MASTER EN INGÉNIEUR DE GESTION À FINALITÉ SPÉCIALISÉE EN DATA SCIENCE

Prédiction de la composition d'une équipe de football sur base d'un modèle de classification

Vermaut, Benjamin

Award date:
2022

Awarding institution:
Universite de Namur

Prédiction de la composition d'une équipe de football sur base d'un modèle de classification

Benjamin VERMAUT

Directeur : Prof. S.FAULKNER

Mémoire présenté en vue de l'obtention du titre de
Master 120 - Ingénieur de gestion
Data Science

ANNÉE ACADÉMIQUE : 2021-2022

Université de Namur, ASBL

Faculté des Sciences économiques, sociales et de gestion – Département des Sciences de gestion

Rempart de la Vierge 8, B-5000 Namur, Belgique, Tel. +32 [0]81 72 49 58/48 41

Remerciements

Premièrement, je souhaiterais remercier mon promoteur, le professeur Faulkner qui m'a permis de choisir ce sujet de mémoire. Ensuite, je souhaiterais remercier l'Union Royale Belge de Football et plus particulièrement Yannick Euvrard, mon maître de stage, pour l'expérience unique qui m'a été offerte lors de mon stage. Finalement, je tiens à remercier StatsBomb de mettre à disposition des données librement accessibles.

Résumé

Beaucoup d'informations sont prises en compte par les entraîneurs lorsqu'ils décident de la composition d'équipe pour un match. Les modèles de machine learning, ayant la capacité de trouver des relations cachées parmi les données, peuvent apporter une source d'information supplémentaire aux analyses fournies par les assistants de l'entraîneur principale. Le modèle développé dans cette thèse, se base sur la méthode de classification supervisée des données. Il a pour but d'aider les entraîneurs dans la décision de la composition d'équipe. Les résultats du modèle sont d'abord analysés de façon standard en évaluant le modèle sur les saisons sur lesquelles il a été entraîné. Il est ensuite également évalué sur la saison 2020/2021 de La Liga, qui est alors inconnue du modèle. Les premiers résultats obtenus par le modèle sont prometteurs pour la classification des joueurs comme devant jouer ou non pendant un match.

Summary

A lot of information is taken into account by coaches when taking the decision on the team composition for a match. Machine learning models, having the ability to find hidden relationships among the data, can provide an additional source of information to the analyses provided by the head coach's assistants. The model developed in this thesis is based on the supervised data classification method. It is intended to assist coaches in making team composition decisions. The results of the model are first analysed in a standard way by evaluating the model over the seasons over which it has been trained. It is then also evaluated on the 2020/2021 La Liga season, which is then unknown to the model. The first results obtained by the model are promising for the classification of players as playing or not playing during a match.

Table des matières

1 Introduction

1.1 Contexte

Un entraîneur doit constamment réfléchir à la composition de son équipe afin de pouvoir battre l'adversaire lors de chaque match. Dans un objectif continu d'amélioration des performances des joueurs d'une équipe de football, les clubs et les coachs ont de plus en plus recours à l'analyse des performances des joueurs. De nombreux clubs recrutent aujourd'hui des analystes de performances, analystes vidéos ou analystes de données pour aider le coach principal dans cette tâche. Ces analyses ont pris une ampleur importante ces dernières années avec le développement des nouvelles technologies telles que la vidéo, l'Internet des objets, le big data, le cloud et plus récemment encore la récolte automatique de données via des caméras intelligentes utilisant du computer vision et des technologies de détection. Néanmoins, la majorité de ces analyses se concentrent actuellement uniquement sur l'analyse statistique des performances passées. Un domaine qui n'est pas encore suffisamment développé est celui de l'analyse prédictive via l'utilisation du machine learning. Le but est de permettre d'étendre l'analyse des performances par des prédictions et des recommandations sur la composition d'équipe à utiliser par l'entraîneur pour un match à venir.

1.2 Question de recherche

Cette thèse se concentre sur la création d'un modèle de machine learning pour prédire la composition d'une équipe de football pour un match. La question que cette thèse cherche à répondre est donc la suivante :

Comment prédire la composition d'une équipe de football pour un match de championnat ?

D'autres sous-questions vont découler de cette question de recherche principale telles que :

— Quel est l'algorithme le plus performant dans la prédiction de la composition d'une équipe de football pour un match de championnat ?

— Quelles sont les variables qui influencent le plus la prédiction de la composition d'une équipe de football pour un match de championnat ?

1.3 Structure de la thèse

Pour répondre à la question de recherche principale, cette thèse est divisée en cinq parties. Premièrement, **une revue de la littérature** existante en matière d'analyse des performances des joueurs de football ou d'une équipe entière. Ensuite, la seconde partie a pour but d'expliquer **la méthodologie** mise en œuvre pour répondre à la question de recherche. Cette partie explique les étapes nécessaires à la création du modèle de machine learning pour prédire la composition d'une équipe de football. La troisième partie se concentre sur **l'analyse des données**. Il y est détaillé la façon dont les données ont été collectées, assemblées et modélisées. Elle présentera également la sélection du modèle optimal. La quatrième partie analyse **les résultats** obtenus par le modèle sélectionné. Enfin la dernière partie présente **une discussion** visant à mettre en perspective les apports de cette thèse avec la littérature existante.

2 Revue de la littérature

Ce chapitre présente différents articles scientifiques liés à la problématique des modèles de prédiction utilisés dans le secteur du football. Il présente d'abord l'origine des analyses des performances dans le milieu du football ainsi que l'évolution récente de la littérature scientifique. Ensuite, il présente la littérature existante dans l'évaluation des performances des joueurs. L'utilisation de la théorie des graphes et de l'analyse de réseaux est par la suite détaillée. Il recense ensuite la littérature existante sur les sujets liés au recrutement et à la composition d'équipe. Enfin, il termine par la présentation d'articles scientifiques qui s'intéressent à la prédiction des résultats des matchs de football.

2.1 Évolution de la littérature scientifique

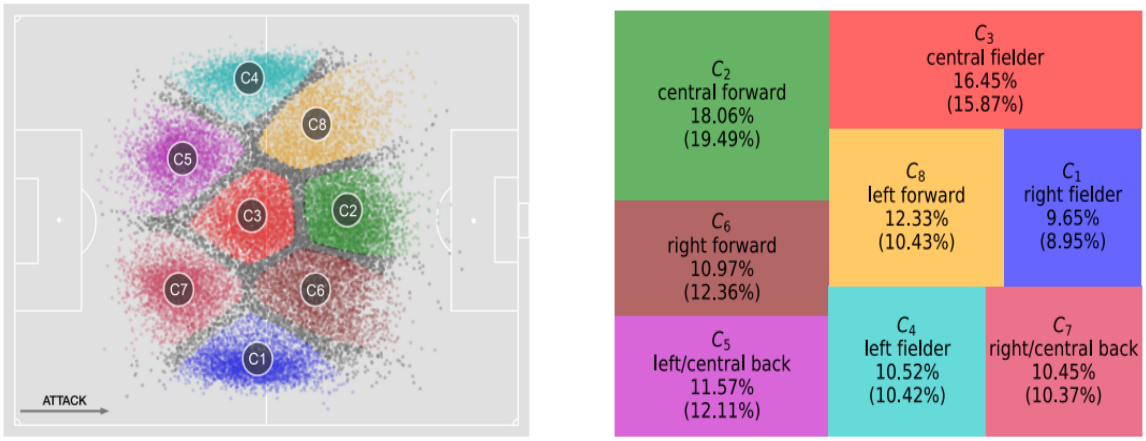
Dans le football, l'histoire des statistiques commence en 1950 lorsque Charles Reep, considéré comme le pionnier de l'analyse notationnelle et des performances, prit un crayon pour noter les statistiques du match entre Swindon Town et Bristol Rovers. Il a continué de collecter des données lors de presque 500 matches entre 1930 et 1960. Il est alors devenu l'un des premiers analystes professionnels du football, comme l'indique Pollard (2002). Ses recherches innovantes sur les probabilités de succès des passes continuent d'impacter le football moderne. Reep et al. (1971) ont montré que les probabilités de succès d'un enchaînement de passes entre les joueurs de football suivent la loi binomiale négative. Il en va de même pour les probabilités de succès de l'enchaînement des passes se terminant par un tir ou un but ainsi que les probabilités associées au nombre de buts marqués par une équipe pendant un match de football.

Il y a eu, ces dernières années, une accélération des innovations en matière de modèles prédictifs dans le football avec le développement des nouvelles technologies telles que la vidéo, l'Internet des objets, le big data, le cloud et plus récemment encore la récolte automatique de données via des caméras intelligentes utilisant du computer vision et des technologies de détection. Gudmundsson and Horton (2016) identifient depuis les années 2005-2010 un accroissement des articles scientifiques concernant l'analyse des données spatiales dans les sports d'équipes qui sont devenues de plus en plus disponibles pour les chercheurs. Ils y décrivent un accroissement des recherches telles que l'étude des représentations des phases de jeu, la cartographie de l'intensité de différents évènements via des heatmaps, la modélisation des déplacements et mouvements les plus récurrents, l'identification des formations d'équipes utilisées, l'analyse des réseaux des équipes, ainsi que le développement de modèle de data mining pour la prédiction d'évènements futurs. De nombreuses recherches ont également contribué à la création de différentes mesures de performances utilisées pour évaluer les performances des joueurs.

2.2 Évaluation des performances des joueurs

L'analyse des performances des joueurs, est actuellement le domaine sur lequel se porte le plus d'attention de la part des analystes des clubs de football. Pappalardo et al. (2019) ont développé a data-driven framework that offers a principled multi-dimensional and role-aware evaluation of the performance of soccer players.

Par l'utilisation de Linear Support Vector Classifier (LSVC), ils sont parvenus à regrouper des joueurs de football selon leurs performances. Chaque groupe représente en réalité un rôle sur le terrain. Ils sont arrivés à identifier 8 rôles sur le terrain comme montré à la figure 2.1a.



(a) (b)

Figure 2.1 – (a) Groupes réalisés autour des centres de performance $C_1, ..., C_8$ tels que découverts par Pappalardo et al. (2019). (b) Distribution des huit positions découverte par Pappalardo et al. (2019).

Ils utilisent ensuite ces huit rôles pour évaluer les performances des joueurs au sein des différents rôles. On peut voir à la figure 2.2 l'évaluation des performances réalisée grâce à Playerank. Cela permet d'avoir une vue claire sur les performances des joueurs par position évaluée grâce au rating, mesure de performance créée par Pappalardo et al. (2019).

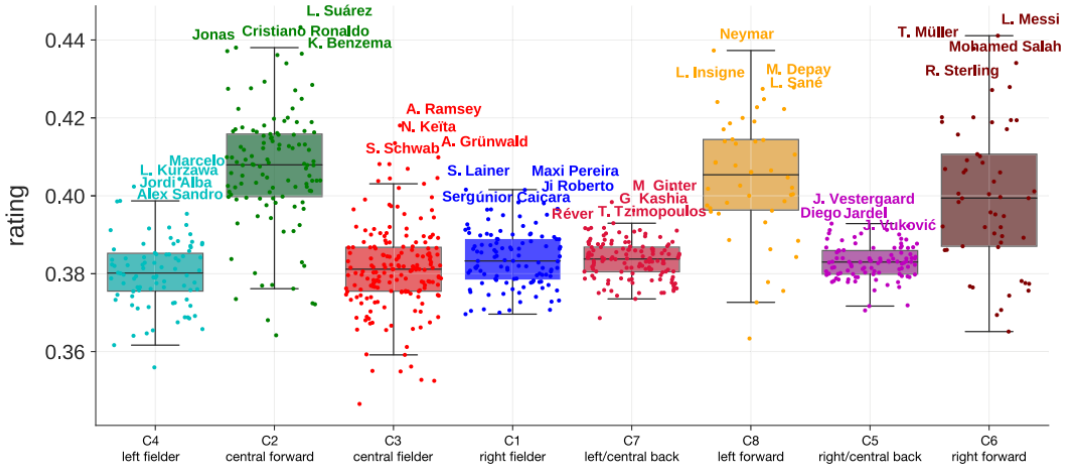


Figure 2.2 – Résultas obtenus par Pappalardo et al. (2019) montrant la distribution de l'évaluation des performances (rating) des joueurs pour chaque position. Chaque diagramme en boîte représente un cluster (position) et chaque point représente les performances du joueur mesuré par le rating.

Pouvoir avoir une liste des performances individuelles de chaque joueur pour chaque position regroupée par une seule métrique, ici le rating, permet d'identifier les meilleurs joueurs pour chaque position selon cette mesure. Cela peut aider l'entraîneur dans le choix de la composition d'équipe pour un match mais également pour identifier des joueurs à surveiller en vue d'un transfert.

Ils ont comparé les résultats de Playerank avec la mesure de Flow Centrality développée par Duch et al. (2010) qui est définie comme le nombre de fois qu'un joueur intervient dans un enchaînement de passe qui se termine par un tir et la mesure Pass Shot Value (PSV) développée par Brooks et al. (2016) qui mesure l'importance des passes dans la création de tirs. Ils ont comparé les résultats fournis par ces trois mesures pour évaluer les performances des joueurs avec des évaluations de ces performances effectuées par des scouts professionnels. Un scout est une personne responsable de l'évaluation de la performance de joueurs d'autres clubs en vue d'un transfert. Ils sont arrivés à la conclusion que les résultats fournis par Playerank sont plus similaires aux résultats des scouts que PC et PSV.

2.3 Théorie des graphes et analyse des réseaux

L'analyse de réseau et la théorie des graphes permettent de comprendre les interactions entre les joueurs, le style de jeu d'une équipe et encore l'importance des différents joueurs dans le schéma tactique d'une équipe de football. L'analyse des réseaux de passes dans les sports de ballons en équipes a été introduite par Passos et al. (2010) qui ont montré qu'elle pouvait servir à identifier des schémas de jeu réguliers. Grund (2012) regarde par exemple la centralité pour identifier les joueurs clés dans le réseau de passes ainsi que l'intensité et la densité du réseau de passes. Ils sont ainsi arrivés à la conclusion qu'une augmentation de la fréquence des passes entraîne l'augmentation des performances de l'équipe et que l'augmentation de la centralisation de l'équipe tend, elle, à diminuer les performances de l'équipe. Cela permet donc aux entraîneurs de comprendre qu'il faut éviter que l'équipe centralise toutes ses passes sur un petit nombre de joueurs et qu'il faut donc privilégier un style de jeu favorisant les passes entre les joueurs. La betweeness centrality peut aussi être analysée pour mesurer l'impact que provoque la substitution d'un joueur dans le réseau de passe. Une équipe devrait pouvoir avoir un réseau de passe suffisamment étendu pour pouvoir ne pas être mis en difficulté lors du remplacement d'un joueur. La closeness centrality peut, elle, être utilisée pour voir à quel point il est facile de joindre un joueur en particulier. Enfin, l'Eigenvector centrality et le PageRank peuvent, eux, être utilisés pour calculer la probabilité qu'un joueur soit en possession de la balle après un certain nombre de passes. Une répartition uniforme tend à montrer qu'il n'y a pas de joueurs qui ont un rôle central dans le schéma de passes. L'analyse de tous ces indicateurs de centralités peut permettre aux analystes d'une équipe de mieux comprendre l'impact de chaque joueur de l'équipe et ainsi de prendre des décisions plus réfléchies lors de la sélection des joueurs pour une rencontre. Mais cela peut aussi servir à analyser l'équipe adverse, son style de jeu et les joueurs indispensables dans le schéma de jeu.

Stöckl et al. (2021) ont développé un modèle utilisant les réseaux de neurones convolutifs pour effectuer des prédictions en temps réel sur base des trackings data, données de traçage en anglais. Ces données permettent d'avoir connaissance des positions ainsi que des déplacements de tous les joueurs sur le terrain. Ils ont, avec leur modèle, développé trois mesures de prédiction en temps réel des évènements qui vont se produire pendant un match. Xpass mesure la probabilité qu'un joueur qui a le ballon réussisse une passe vers un coéquipier. Xreceiver mesure la probabilité de chaque joueur à recevoir la balle via une passe d'un de ses coéquipiers. Xthreat, mesure la probabilité qu'un attaquant effectue un tir dans les 10 secondes après avoir reçu le ballon. Ces mesures peuvent ensuite être utilisées pour représenter des cartes de chaleurs des performances moyennes d'un joueur pendant un match. On peut ainsi voir les zones du terrain ou le joueur a été le plus dangereux pendant le match et les comparer avec d'autres joueurs.

De plus, ces mesures permettent de fournir des informations supplémentaires lors de l'analyse vidéo des matchs.

2.4 Le recrutement et la composition d'équipe

Les clubs de football consacrent des efforts importants dans le recrutement de nouveaux joueurs. Ils emploient de nombreux scouts à travers le monde pour identifier et surveiller de potentiels joueurs qui pourraient être intéressants pour le club. Vu le nombre de joueur de football dans le monde, il est intéressant de pouvoir utiliser des modèles basés sur les données pour aider les clubs à détecter des nouvelles pépites.

Payyappalli and Zhuang (2019) ont notamment développé un modèle basé sur les données pour la prise de décision des clubs de football sur les transferts. Ce modèle se concentre sur l'évolution de la valeur des joueurs au fil du temps pour évaluer le succès d'un transfert. Ainsi, il est plutôt axé sur la rentabilité des transferts et son objectif est de maximiser l'utilité du joueur pour le club. Bien que d'autres variables rentrent en jeu, telles que l'âge du joueur, sa nationalité ou encore sa position, la valeur d'un joueur reste étroitement corrélée à ses performances. Pour développer leur modèle, ils ont utilisé les données de plus de 7000 joueurs de Premier League, le championnat de première division anglaise, issues du site web Sofifa.com. Ils ont ensuite utilisé ces données pour optimiser leur modèle d'optimisation qui utilise la méthode de la moyenne mobile simple pour prédire la valeur et le salaire futur du joueur. Il est très intéressant de remarquer qu'ils intègrent à leur modèle les différentes contraintes auxquelles un club doit se plier telles que les régulations de la Premier League. Ils montrent que leur modèle aurait permis aux 12 clubs de Premier League sur lesquels ils ont testé le modèle, d'augmenter la valeur marchande globale de leurs équipes s'ils avaient suivi les recommandations fournies par le modèle. On peut voir à la figure 2.3 un exemple d'augmentation de la valeur marchande par le modèle pour quatre clubs de Premier League.

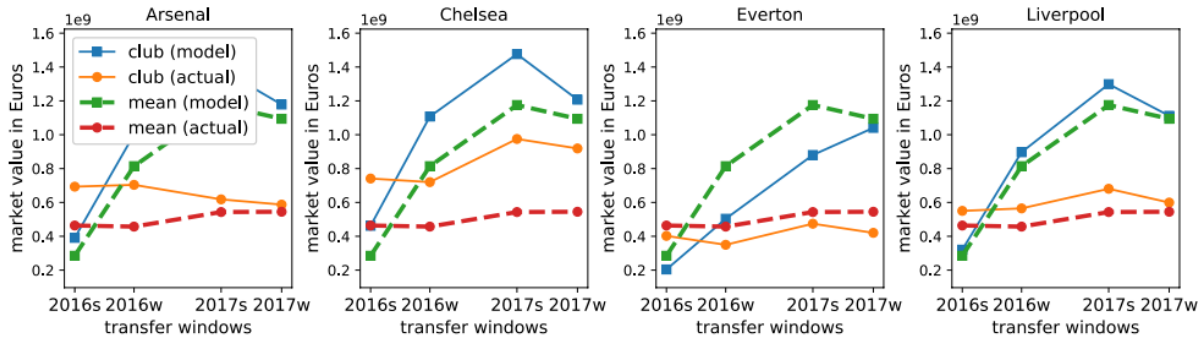


Figure 2.3 – Comparaison de la valeur marchande actuelle et prédite par le modèle d'Arsenal, Chelsea, Everton et Liverpool au cours de quatre périodes de transferts, obtenue par Payyappalli and Zhuang (2019) en utilisant la méthode de la moyenne mobile simple.

Il est cependant dommage que cet article ne montre pas d'exemple de joueurs qui ont été conseillés par le modèle pour un club de football. À l'inverse, Young and Weckman (2020) ont eux créé un système d'aide à la décision sur la compatibilité avec l'équipe qui renvoie une liste concrète de joueurs qui devraient être transféré pour chaque équipe de National Football League. Bien que cet article se concentre sur le football américain, de nombreux parallélismes peuvent être effectués avec le football et avec la question de recherche de cette thèse. Ils proposent un

modèle mathématique qui utilise le machine learning et les méthodologies statistiques pour aider les entraîneurs dans le recrutement de nouveaux joueurs.

Ce qui est très intéressant avec ce modèle est qu'il optimise le transfert des joueurs dans le but de maximiser les performances de l'équipe. Pour pouvoir calculer une estimation des performances de l'équipe, ils ont adapté la formule développée par Bill James dans les années 80, qui permet d'estimer le pourcentage de victoire d'une équipe pour la ligue majeure de baseball (MLB).James (2003) montre dans la quatrième réédition de son livre, comment il a créé la formule du Pythagorean Winning percentage en partant de l'hypothèse simple "qu'une équipe gagnerait plus de la moitié de leurs matchs s'ils étaient capables de marquer plus de points que de points encaissés".

$$PythagoreanWinning\% = \frac{RunsScored^x}{RunsScored^x + RunsAllowed^x}$$

À l'époque Bill James assignait à x la valeur de 2. Mais Young and Weckman (2020) ont recalculé x en minimisant l'erreur quarrée entre les pourcentages de victoires réels et estimés par un modèle qu'ils ont créés. Pour ce faire, ils ont collecté les données de Pro-Football Reference (2009) de 1988 à 2007 qui leur permet d'avoir des données sur les points marqués et les points encaissés. Ils ont finalement obtenu un x estimé par la méthode des moindres carrés équivalente à 2,49. Ce x estimé appliqué à la formule du Pythagorean winning % leur permettait de prédire le pourcentage de victoire d'une équipe avec un coefficient de détermination de 85,2% et un intervalle de confiance de $\pm$0,147 ce qui correspondait à $\pm$ 2,36 matchs d'une saison régulière de NFL.

Pour leurs données d'entrées ils ont utilisé les données du jeu vidéo Madden NFL ainsi que les données présentes dans Sports Xchange qui recense les performances des joueurs lors de la NFL Combine. Il s'agit d'un évènement d'une semaine durant lequel les athlètes des ligues universitaires réalisent des tests physiques et psychologiques pour espérer taper dans l'œil d'entraîneurs et de scouts de la NFL.

Pour prédire les performances d'une équipe pour une saison, ils ont calculé le pythagorean winning percentage sur base des données d'entrées des joueurs de la saison précédente. Ils ont pour ce faire utilisé trois modèles de réseaux de neurones artificiels. Ils calculent également l'impact d'un nouveau joueur dans ce pourcentage de victoire en remplaçant un joueur de l'équipe actuelle par ce joueur. En comparant la différence entre les deux pythagorean winning percentage, ils déterminent s'il est bénéfique de le transférer ou non. Ils vont même jusqu'à distinguer la contribution réelle et théorique qu'un nouveau joueur peut apporter. Ils sont partis du constat simple qu'un joueur de NFL pouvait être blessé pendant une partie des matchs d'une saison. Ainsi, la theoretical contribution value va tenir compte des absences du joueur en estimant quelles auraient été ses performances lors des matchs qu'il n'a pas joués. Elle permet donc de prédire l'impact d'un joueur s'il avait joué tous les matchs d'une saison. À l'inverse, l'expected contribution value ne cherche pas à estimer les performances des joueurs lorsqu'ils n'ont pas joué et aura donc des performances équivalentes à 0 lorsqu'il n'a pas joué.

On peut voir à la figure 2.4 un exemple des résultats qu'ils obtiennent avec la méthodologie *Heuristic Evaluation of Artificially Replaced Teammates (HEART)* développée par Young and Weckman (2020) pour l'équipe des Cleveland Browns juste avant le début de la saison 2007. Pour chaque réseau de neurones artificiels (A, B, C) utilisé avec la méthodologie HEART, ils ont prédit le Baseline (win%) qui correspond au Pythagorean winning percentage de l'équipe de base de 2007 sans effectuer le moindre changement de joueur. Ensuite, ils remplacent Leigh

Bodden qui était le CornerBack des Cleveland Browns par Darrelle Revis, pouvant alors être sélectionné par n'importe quelle équipe lors de la Draft de 2007. Ils calculent le pythagorean winning percentage des Cleveland Browns en ayant remplacé Bodden par Revis et obtiennent un Expected Replacement (win%). Pour le modèle A, le remplacement de Browns par Revis permettrait aux Cleveland Browns d'atteindre un pourcentage de victoires prédit de 60,9% ce qui augmenterait le pythagorean winning percentage de 9,7% en se basant sur l'expected contribution value. C'est ce que montre l'Expected Delta (win%). Ils effectuent le même calcul pour la theoretical contribution value et obtiennent, avec le modèle A, une augmentation du winning percentage de 20,4%, passant de 51,2% à 71,6%.

HEART Model	Team City	Baseline (win%)	Expected Replacement (win%)	Expected Delta (win%)	Theoretical Replacement (win%)	Theoretical Delta (win%)
A	Cleveland	51.2	60.9	9.7	71.6	20.4
B	Cleveland	47.7	47.5	-0.2	47.9	0.1
C	Cleveland	53.4	47.6	-5.8	61.2	7.8

Figure 2.4 – Example d'expected et theoretical contribution value obtenues par Young and Weckman (2020) en utilisant la méthodologie HEART.

Il est intéressant de noter que les résultats varient assez fortement d'un modèle à l'autre, surtout pour le theoretical replacement (win

Ce qui est donc très intéressant avec ce modèle est qu'il est possible de voir les joueurs qui ont les meilleures expected et theoretical contribution value pour une équipe en particulier. On peut voir à la figure 2.5 un exemple des joueurs qui ont les meilleures expected et theoretical contribution value pour l'équipe des Oakland Raiders de 2007. Ainsi, on y voit que selon la méthodologie HEART, Patrick Willis est le joueur qui a l'expected et la theoretical contribution, remise ici en nombre de matchs supplémentaires gagnés prédis, les plus élevées. Il permettrait de faire gagner de 3 à 3,27 matchs supplémentaires selon les mesures utilisées. On voit également l'ordre réel auquel chaque joueur a été choisi lors de la Draft de 2007 toutes équipes confondues. On voit donc que JaMarcus Russel qui était 4ème dans les recommandations du modèle HEART a été sélectionné en 1ère position lors de la Draft de 2007.

Les résultats obtenus par la méthodologie HEART sont très intéressants parce qu'ils peuvent s'adresser directement aux scouts en charge des transferts de nouveaux joueurs et aux managers d'une équipe. Et bien que cette méthodologie se concentre sur le football américain, il serait intéressant de l'adapter au football. Beaucoup d'équipes de football seraient intéressées par un modèle basé sur les données qui permettrait de fournir une liste des joueurs qui auraient le plus d'impact sur le nombre de victoires prédites d'une équipe.

Néanmoins, il est extrêmement important de rappeler qu'il s'agit de prédictions basées uniquement sur des données. En analysant la liste des meilleurs joueurs selon l'expected et la theoretical contribution value fournies par la méthodologie HEART, Young and Weckman (2020) se sont rendus compte que le modèle avait permis de détecter des futurs grands joueurs qui avaient été sélectionnés parmi les derniers lors de la Draft de 2007. Mais également que certaines équipes qui avaient choisi des joueurs différents que ceux prédis par HEART avaient sélectionné des joueurs qui ont eu un palmarès beaucoup plus impressionnant que les joueurs alors conseillés

Pos.	Name (Last, First)	Act. Draft Selection	Expected (games)	Theoretical (games)	Expected Rank	Theoretical Rank
LB	Willis, Patrick	11	3.00	3.27	1	1
RB	Peterson, Adrian	7	2.53	3.03	2	2
QB	Quinn, Brady	22	2.40	2.43	3	3
QB	Russell, JaMarcus	1	2.23	2.27	4	4
CB	Revis, Darrelle	14	1.23	1.90	7	5

Figure 2.5 – exemple des joueurs ayant les expected et theoretical contribution value les plus élevées obtenues par Young and Weckman (2020) en utilisant la méthodologie HEART pour l'équipe des Oakland Raiders de 2007.

par HEART. Certains joueurs qui ont été conseillés par HEART se sont également avérés ne pas avoir eu une belle carrière. Cela reste un sport collectif avec son lot d'incertitudes. Et que ce soit dans le football américain ou dans le football, il existe de nombreux exemples de joueurs qui ont étés achetés par des clubs comme étant des futures stars et qui n'ont pas eu par la suite les performances espérées. À l'inverse, il y a également beaucoup d'exemples de joueurs dont certains clubs ne voulaient plus et qui ont eu des performances incroyables dans leurs nouveaux clubs.

2.5 Prédiction des résultats d'un match

Baboota and Kaur (2018) ont développé un modèle de machine learning visant à prédire les résultats des matchs de football d'English Premier League. Pour la réalisation de leur modèle de machine learning, ils ont utilisé des indicatifs de performances des équipes. Ils ont notamment utilisés les statistiques évaluant les performances de l'attaque, du milieu de terrain et de la défense ainsi que la performance générale de l'équipe issue de `https://www.fifaindex.com/teams/`. Ces statistiques sont créées par l'algorithme du jeu vidéo FIFA et ont pour but de refléter la force des équipes dans les différents secteurs de jeu. Constantinou and Fenton (2013) se sont rendus compte que la différence de buts, qui représente la différence entre le nombre de buts marqués et le nombre de buts encaissés à un moment t, jouait un rôle important dans la prédiction du résultat d'un match. Ils ont également essayé d'utiliser une mesure permettant de calculer l'état de forme récent d'une équipe pour prédire le résultat d'un match. Ils ont créé cette mesure en assignant une valeur aux résultats qui dépend de la distance temporelle avec la date d'un match. Les résultats les plus anciens ayant moins de valeur que les résultats les plus récents. Il est intéressant de noter que cela a résulté en une limitation de leur modèle. Ils se sont aperçus qu'il était de plus en plus performant au fil de la saison et qu'il n'était pas très performant pour les premiers matchs de la saison.

Goes et al. (2021) montrent qu'ils arrivent à prédire avec une précision de presque 75% le résultat d'un match en utilisant l'algorithme de l'analyse discriminante linéaire (LDA). Ils se basent sur les données de positions des joueurs pendant un match pour entraîner leur modèle dans la classification d'un match selon qu'il s'agisse d'une victoire ou d'une défaite. La figure 2.6 montre l'évolution des probabilités de victoire des deux équipes au fil d'un match.

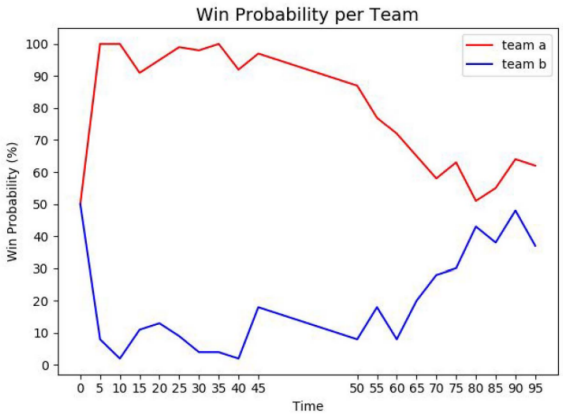


Figure 2.6 – Distribution des probabilités de victoire de deux équipes obtenue par le modèle d'analyse discriminante linéaire développé par Goes et al. (2021) en utilisant les données de positions des joueurs.

De manière similaire, Robberechts et al. (2021) montrent la probabilité de victoire à tout moment de chaque équipe lors d'un match de football. Ils ont utilisé un modèle de classification naïve bayésienne. Contrairement à Goes et al. (2021), ils n'ont pas utilisé les données de positions des joueurs. Ils se sont plutôt orientés vers des données événementielles. Ainsi, ils ont pris en compte des données tel que le temps de jeu restant, la différence de score, le nombre de buts marqués, le nombre de cartons jaunes et rouges reçus, le nombre d'opportunités de but, le pourcentage de duels gagnés, le nombre de passes offensives, l'Expected Threat (Xthreat) développée entre autres par Stöckl et al. (2021) ainsi que la différence d'ELO entre les deux équipes. Tirée du monde des échecs, l'ELO est une mesure qui représente la différence de niveau entre deux équipes. Il s'agit d'un classement attribué à chaque équipe selon les niveaux de performances de celle-ci. L'utilité de l'utilisation de l'ELO dans la prédiction de victoire d'une équipe a été démontrée par Hvattum and Arntzen (2010).

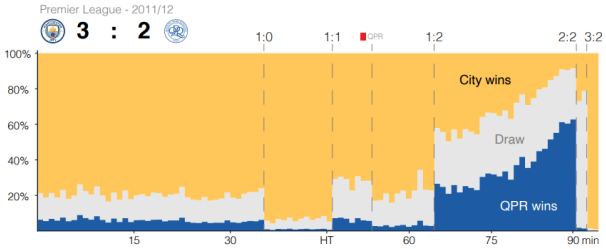


Figure 2.7 – Distribution des probabilités de victoire de Manchester City et des Queens Park Rangers (QPR) obtenue par le modèle de classification naïve bayésienne développé par Robberechts et al. (2021).

3 Méthodologie

Ce chapitre explique les étapes nécessaires à la création du modèle de machine learning pour prédire la composition d'une équipe de football. Il commence par une introduction au machine learning et aux algorithmes de classification y sera faite. Ensuite, cette section explique les différentes étapes de prétraitement des données utiles à l'amélioration des performances du modèle de classification. Enfin, il y sera détaillé les différentes mesures utiles à l'évaluation des performances d'un modèle de classification.

3.1 Introduction au Machine Learning

Pour répondre à la question de recherche principale de cette thèse, qui vise à savoir comment on peut prédire les joueurs de football qui vont être sélectionnés pour un match de championnat, il faut tout d'abord passer en revue les différents types d'algorithmes de machine learning qui existent.

Murdoch et al. (2019) définissent les modèles de machine learning comme des modèles capables de produire des connaissances sur les relations du domaine contenues dans les données.

Les modèles de machine learning ont deux objectifs :

— Prédire : Effectuer des prédictions sur des données encore inconnues.

— Décrire : Tirer des informations des données observées.

Le Machine Learning, peut être divisé en deux grandes catégories que sont le Machine Learning Supervisé et le Machine Learning Non Supervisé. Le Machine Learning Supervisé, peut être défini comme une approche de machine learning visant à prédire ou classifier des résultats en se basant sur des données d'entrée et de sorties qui sont étiquetées. Les données d'entrées sont les données fournies à l'algorithme comme sources d'information pour pouvoir effectuer les prédictions recherchées. Cela signifie que l'algorithme travaille sur des données dont il connaît à l'avance le résultat final. Le but du Machine Learning supervisé est de s'entraîner sur des ensembles de données pour pouvoir ensuite être capable de prédire les résultats sur de nouvelles données. À l'inverse, le Machine Learning non supervisé, lui, ne connaît pas à l'avance le résultat final des données d'entrées. Son but est de découvrir des relations dans les données que l'on ne connaît pas à l'avance.

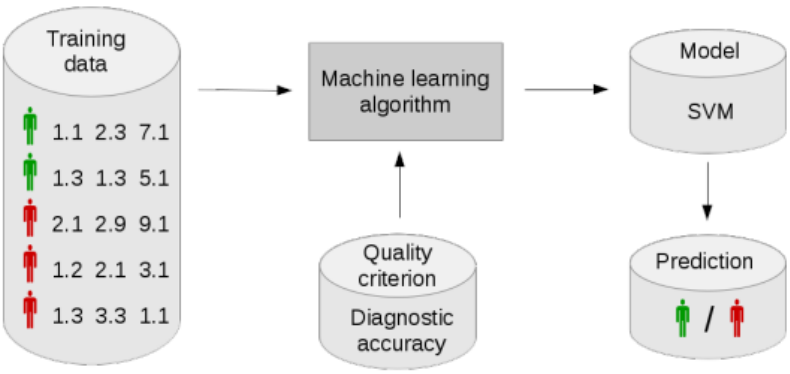


Figure 3.1 – Schéma de synthèse du fonctionnement du Machine Learning supervisé.

Dans le contexte de cette thèse, nous rentrons dans le cadre du Machine Learning Supervisé puisque nous possédons, dans les données, l'information que nous cherchons à prédire, à savoir le fait qu'un joueur ait joué ou non pendant un match de football. Comme montré à la figure 3.1, les algorithmes de Machine Learning utilisent des ensembles de données pour pouvoir s'entraîner et apprendre à prédire un résultat final. Les ensembles de données fournis aux algorithmes doivent être dans la structure suivante :

$$X_{k\times n} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k1} & x_{k2} & \cdots & x_{kn} \end{bmatrix}, y_k = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix}$$

Figure 3.2 – La matrice d'input X et la variable cible y où k est le nombre de lignes du dataset et n le nombre de colonnes.

Dans le cadre de cette thèse, nous utiliserons comme variable cible y la variable contenant l'information quant au fait qu'un joueur ait ou non joué dans les différents matchs de football d'une saison de championnat. On appellera cette variable *played* dans la section 4. Elle indiquera si un joueur a joué (1) ou non (0) lors d'un match. Cette information sera extraite grâce aux données fournies par l'Open Data de StatsBomb. Les open data sont des ensembles de données dont l'accès est public et l'usage est laissé libre à l'utilisateur. StatsBomb est parmi les leaders des fournisseurs de données dans le secteur du football. Ils collectent les données relatives aux matchs de football qu'ils fournissent ensuite à leurs clients. Ils proposent également des logiciels de consultance et d'analyse de ces données. Ils comptent parmi leurs clients des équipes au top niveau international telles que l'AS Roma, le Borussia Dortmund, Villarreal CF ou encore des sélections nationales comme l'Italie ou la Belgique.

La matrice X contiendra, elle, les variables d'entrées qui vont permettre à l'algorithme d'avoir des informations pour prédire la variable cible. Le choix de ces variables d'entrées sera explicité à la section 4. Ces variables d'entrées proviennent de la base de données extraite du jeu vidéo FIFA développé et édité annuellement depuis 1993 par EA Sports, l'un des principaux producteurs et développeurs mondiaux de jeux vidéo. Il s'agit du jeu vidéo de football de référence dans le secteur des jeux vidéo. Il fut le jeu vidéo le plus vendu en France en 2021 avec plus de 1,5 million d'exemplaires vendus. Il l'était également en 2019, 2018 et 2017. Lorsque l'on parle d'une édition de FIFA, on y associe toujours un nombre correspondant à la saison sur laquelle il se base. Cependant, ce nombre ne correspond pas à l'année de sortie mais plutôt à la deuxième année calendaire sur laquelle une saison de football s'étend. Ainsi, FIFA 20 correspond à l'édition qui se base sur la saison 2019-2020 et a été distribué vers la fin septembre 2019.

3.2 Algorithmes de machine learning

Une fois les données correctement structurées, il nous faudra choisir l'algorithme de machine learning adéquat pour prédire au mieux notre résultat. Pour ce faire, il faut d'abord distinguer deux types d'algorithmes de machine learning supervisé. Premièrement les algorithmes de **régression** qui visent à comprendre la relation entre une variable dépendante y et une ou plusieurs variables indépendantes comprises dans la matrice X. Les algorithmes de régression sont utilisés lorsque la variable cible contient des valeurs numériques continues. On les utilisera par exemple pour prédire le prix d'une maison.

En parallèle, on retrouve les algorithmes de **classification** qui permettent, eux, de classifier les données selon différentes catégories. Ils sont privilégiés lorsque les variables cibles sont des variables binaires ou catégorielles, identifiant des catégories spécifiques. On les utilisera par exemple pour différencier un chien d'un chat. La variable cible de notre modèle, *played*, est une variable binaire qui indique si oui(1) ou non(0) un joueur a joué pendant un match de football. On se tournera donc vers l'utilisation d'algorithmes de classification.

Il y a différents types d'algorithmes de classification. Parmi les plus courants on retrouve la régression logistique, l'Analyse discriminante linéaire, la méthode des k plus proches voisins(KNN), les arbres de décision, la classification naïve bayésienne ainsi que les machines à vecteur de support.

3.2.1 Régression Logistique

Pour rappel, F_X est la fonction de répartition d'une variable aléatoire X associe à tout réel x la probabilité d'obtenir une valeur inférieure ou égale :

$$F_X(x) = P(X \leq x)$$

La régression logistique est un modèle statistique linéaire qui est utilisé pour la classification binaire. La fonction sigmoïde qui représente la fonction de répartition de la loi logistique est utilisée pour définir une limite de décision capable de classifier les valeurs de la variable cible (y). La figure 3.3 montre un exemple de fonction sigmoïde utilisée pour classifier les données.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Autrement dit, on peut calculer la probabilité conditionnelle que y appartienne à la classe 1 en sachant X comme étant :

$$p(y = 1|x) = \frac{1}{1 + e^{-w^T x - w_0}}$$

où x correspond à la matrice de variables et w la matrice des paramètres.

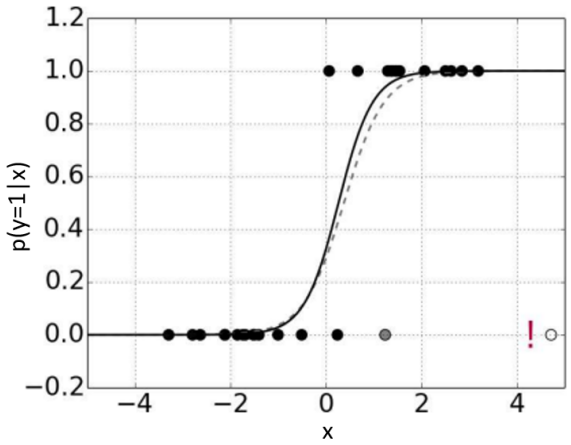


Figure 3.3 – Classification à l'aide de la régression logistique

3.2.2 Analyse Discriminante Linéaire

L'analyse discriminante linéaire est une technique qui permet de classifier les données en groupes distincts. Cette classification est effectuée en trouvant les combinaisons de variables qui permettent de donner la meilleure séparation entre les groupes. L'algorithme va chercher à maximiser la dispersion entre les groupes par rapport à la dispersion à l'intérieur des groupes. On va maximiser **la dispersion entre les groupes** en maximisant :

$$w^T(m_2 - m_1)$$

où

$$m_1 = \frac{1}{N_1}\sum_{n\in C_1} x_n, m_2 = \frac{1}{N_2}\sum_{n\in C_2} x_n$$

m_1 est donc la moyenne de toutes les entrées appartenant à la classe 1 et m_2 la moyenne de toutes les entrées appartenant à la classe 2.

Mais il faut aussi veiller à minimiser les **variances intra-classe**. Pour ce faire, on va minimiser :

$$s_k^2 = \sum_{n\in C_k}(y_n - m_k)^2$$

où $y_n = w^T X, w_k = w^T m_k$

Ainsi, en combinant ces deux conditions, on obtient la formule que cherche à maximiser l'algorithme d'analyse discriminante linéaire :

$$J(n) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$$

On peut montrer que :

$$w \propto S_W^{-1}(m1 - m2)$$

où S_W, la matrice de covariance intra-classe, est :

$$S_W = \sum_{n\in C_1}(x_n - m_1)(x_n - m_1)^T + \sum_{n\in C_2}(x_n - m_2)(x_n - m_2)^T$$

On peut voir à la figure 3.4 l'intérêt de minimiser la distance intra-classe. Cela permet d'obtenir une meilleure fonction de décision et donc une meilleure classification des données dans les différentes classes. Dans le graphique de gauche, on ne considère que la moyenne des classes. On voit donc que la fonction de décision, qui sera approximativement similaire à la droite verte perpendiculaire à l'histogramme, est moins performante pour classifier les données que la fonction de décision du graphique de droite.

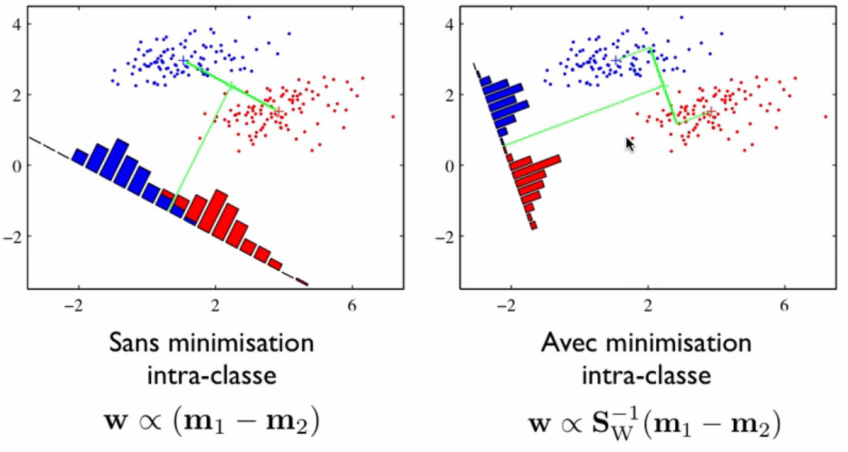


Figure 3.4 – Classification à l'aide de l'analyse discriminante lineaire

3.2.3 Méthode des k plus proches voisins (KNN)

L'algorithme des k plus proches voisins suppose que des objets similaires sont proches les uns des autres. Ainsi, il va chercher à classifier les données en fonction du plus grand nombre de voisins qui appartiennent aux différentes classes.

Tout d'abord, il faut déterminer le nombre de voisins qui vont être utilisés pour classifier la nouvelle instance. Mais avant, il faut comprendre comment est calculée la distance entre les points. Il se base sur le principe de la distance euclidienne pour calculer cette distance et déterminer quels vont être les voisins. Considérons deux points p et p' de coordonnées respectives (x, y) et (x', y'). Leur distance euclidienne peut être calculée comme ceci :

$$||p - p'|| = \sqrt{(x - x')^2 + (y - y')^2}$$

Ainsi, lorsque l'on choisit le nombre de voisins k, on permet à l'algorithme de considérer les k voisins les plus proches selon la distance euclidienne. Une fois ces k voisins sélectionnés, l'algorithme va regarder le nombre de ces voisins qui appartiennent à chaque classe. Il associera à la nouvelle instance d'entrée la classe pour laquelle il y a le plus grand nombre de voisins qui y appartiennent aussi parmi les k voisins. On peut voir à la figure 3.5 à quel point le paramètre k peut influencer la classe à laquelle appartient la nouvelle instance. Avec k=3, l'algorithme KNN classifie la nouvelle instance comme appartenant à la classe B. Alors qu'avec un nombre de 7 voisins, l'algorithme associe la nouvelle instance dans la classe A. Pour déterminer le nombre k optimal, il faudra exécuter plusieurs fois l'algorithme KNN et choisir le k qui permet de réduire le nombre d'erreurs de prédictions.

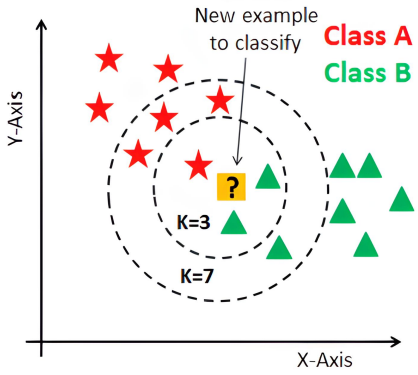


Figure 3.5 – Classification à l'aide de la méthode des k plus proches voisins.

3.2.4 Arbre de décision

Les arbres de décision sont utilisés dans le domaine de l'aide à la décision. Ils modélisent la solution du problème de classification comme une suite de décisions à prendre pour identifier correctement les différentes classes.

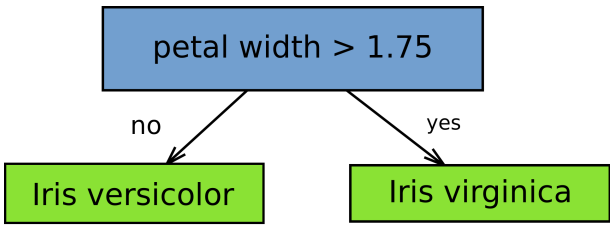


Figure 3.6 – Classification de base à l'aide d'un arbre de décision.

Comme montré à la figure 3.6, le nœud en bleu correspond à une décision à prendre : la largeur des pétales est-elle supérieure à 1.75 ? Si oui, l'algorithme classifiera l'instance comme appartenant à la classe Iris versicolor. À l'inverse, si elle est inférieure à 1.75, l'instance sera classifiée comme appartenant à la classe Iris virginica.

Pour déterminer les nœuds de décision qui vont être sélectionnés, l'algorithme se base sur le score de Gini. Compris entre 0 et 1, le score de Gini, indique la probabilité que l'arbre se trompe lors d'une décision et permet donc d'évaluer la qualité d'une branche. L'algorithme rajoutera à l'arbre de décision la branche possédant le score de Gini le plus élevé. Le score de Gini se calcule comme suit :

$$Gini = \sum p_k * (1 - p_k)$$

où p_k est la probabilité d'obtenir la classe k.

Au départ l'arbre est complètement vide. Ensuite, on évalue le score de Gini de tous les nœuds de décisions potentiels pour ne garder que celui ayant le score de Gini le plus élevé. Et ainsi de suite tant qu'il est possible de diviser les groupes et que le critère d'arrêt n'est pas rencontré. Celui-ci peut être que la profondeur maximale de l'arbre, définie par l'utilisateur, est atteinte ou encore que le nombre d'instances minimum par groupe ne puisse être respecté en effectuant une division supplémentaire du groupe.

3.2.5 Classification Naïve Bayésienne

L'algorithme de classification naïve bayésienne est défini par Saritas and Yasar (2019) comme étant un classificateur de probabilité simple qui calcule un ensemble de probabilités en comptant la fréquence et les combinaisons de valeurs dans un ensemble de données donné. Il se base sur le théorème des probabilités conditionnelles de Bayes :

$$p\left(y|x\right) = \frac{p\left(x|y\right)p\left(y\right)}{p\left(x\right)}$$

Pour rappel, on lit le terme $p\left(y|x\right)$ comme étant la probabilité que l'évènement y se réalise en sachant que x s'est déjà réalisé. L'algorithme de classification naïve bayésienne suppose que toutes les variables sont indépendantes les unes des autres. Son adjectif "naïf" tient son origine de cette hypothèse forte qui n'est pas souvent valide dans le monde réel. Néanmoins, il n'en reste pas moins très utile pour classifier des données en différents groupes. Concrètement, l'algorithme va calculer la probabilité conditionnelle ($p\left(y|x\right)$) que l'évènement du vecteur cible y se réalise en connaissant les valeurs de la matrice d'entrée x correspondante. Prenons par exemple :

— $p\left(joue|salaire\_eleve, jeune, selection\_nationale\right)$: la probabilité qu'un joueur de football joue pendant un match en sachant qu'il a un salaire élevé, qu'il est jeune et qu'il ait déjà été sélectionné en équipe nationale.

— $p\left(jouepas|salaire\_eleve, jeune, selection\_nationale\right)$: la probabilité qu'un joueur de football ne joue pas pendant un match en sachant qu'il a un salaire élevé, qu'il est jeune et qu'il ait déjà été sélectionné en équipe nationale.

L'algorithme classifiera le nouveau joueur, encore "inconnu", selon la classe avec la probabilité la plus élevée au regard des caractéristiques du joueur (représentées par la matrice x).

3.2.6 Machine à vecteur de support (SVM)

Les machines à vecteur de support, également appelées séparateurs à vaste marge reposent sur deux concepts clés : le principe de **marge maximale** et celui de **l'optimisation quadratique**.

Comme montré à la figure 3.7, l'algorithme à vecteur de support cherche à maximiser la distance entre la frontière de décision $h(x) = w^T x + w_0$ et les vecteurs de support correspondants aux instances x d'une classe les plus proches des instances de l'autre classe. Autrement dit, il s'agit de la plus grande distance qui permet de séparer les deux classes de manière homogène. On appelle cette distance la marge.

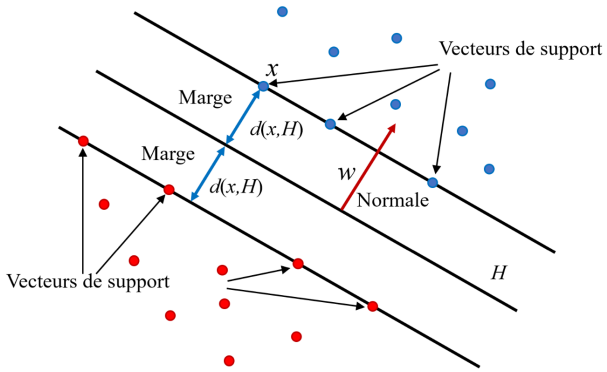


Figure 3.7 – Classification à l'aide d'une machine à vecteur de support.

L'algorithme de machine à vecteurs de support permet de résoudre des problèmes non linéaires. Pour ce faire, le SVM va augmenter l'espace de représentation de X en un plus grand espace où en espérant qu'il existe une séparation linéaire dans ce plus grand espace. Autrement dit, il va augmenter le nombre de dimensions de la matrice X jusqu'à permettre une séparation linéaire des données. Cette transformation va être effectuée grâce à la fonction de noyau. Différentes fonctions de noyau peuvent être utilisées selon les situations. Cependant, la fonction de noyau à base radiale (RBF) est souvent utilisée et se calcule comme suit :

$$K(x, x') = e^{-\gamma||x-x'||^2}$$

où $\gamma = \frac{1}{n\_variables * \sigma^2}$ et σ est la variance et est obtenu par validation croisée sur les données de l'ensemble d'entraînement.

On peut voir à la figure 3.8, l'application de la fonction à base radiale (RBF) pour transformer un plan à deux dimensions en un plan à plus haute dimension permettant une séparation linéaire des données en deux classes.

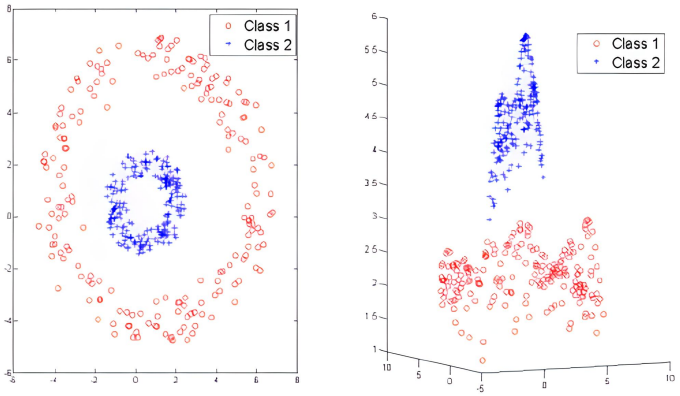


Figure 3.8 – Exemple d'un noyau de fonction à base radiale (RBF) mettant en correspondance des données d'un espace séparable non linéaire à un espace séparable à haute dimension.

L'algorithme SVM possède, en plus de l'hyperparamètre gamma, un hyperparamètre C. Également appelé paramètre de régularisation, il permet de contrôler le taux d'erreur acceptable lors de la classification des données pendant l'entraînement du modèle dans le but d'optimiser l'erreur de généralisation. Il s'agit de la différence de performance entre les performances du modèle sur les données d'entraînement et sur les données de validation.

3.3 Pré traitement

Pour pouvoir évaluer les performances d'un modèle de machine learning, il faut toujours tester le modèle sur des nouvelles données encore inconnues. Ainsi, on doit toujours séparer les données en deux groupes distincts :

— les **données d'entraînement** qui sont utilisées pour l'apprentissage du modèle.

— les **données de validation** qui sont utilisées pour évaluer les performances obtenues par le modèle en comparant les résultats prédits par le modèle avec les valeurs cibles effectives.

Par défaut scikit-learn, la bibliothèque de référence pour l'utilisation de machine learning en Python, défini la répartition des données comme étant 75% des données pour l'entraînement et 25% des données pour la validation. Néanmoins, la répartition idéale des données d'entraînement et de validation varie selon la taille du jeu de données. Une répartition entraînement/test de 80/20% ou même de 90/10% est recommandée pour les larges jeux de données. À l'inverse, si le jeu de données est plus petit, une répartition 60/40% à 70/30% sera préférée. L'utilisateur doit donc choisir arbitrairement une valeur pour la répartition des données d'entraînement et de test. Néanmoins, l'utilisateur pourrait exécuter les différents algorithmes pour différentes valeurs de répartition des données et choisir celle qui donne les meilleures performances.

Cependant, en séparant les données une seule fois avec la méthode expliquée ci-avant, on empêche certaines données de pouvoir être utilisées pour entraîner le modèle. Il en va de même pour les données de validation. Pour pouvoir évaluer la performance d'un modèle, il faudrait pouvoir utiliser toutes les données à la fois pour l'entraînement du modèle et pour sa validation. C'est ce que permet de réaliser la méthode de la **k-fold cross-validation**. Il s'agit de la même méthode que celle expliquée ci-avant mais simplement répétée plusieurs fois. Ainsi, le dataset est divisé en k plis de tailles identiques. La valeur de k, défini par l'utilisateur, ne varie pas lors des répétitions. Pour chacun des k plis, on entraîne le modèle en utilisant un jeu de données d'entraînement équivalent à $k-1$ plis et on utilise le plis restant comme jeu de données de test. On calcule une mesure de performance du modèle. Les différentes mesures de performance seront explicitées à la section 3.4. Les plis du jeu de données ne peuvent être utilisés qu'une seule fois comme jeu de données de validation. Avant chaque répétition le jeu de données est remélangé de sorte que toutes les données puissent être réparties à nouveau en donnée d'entraînement ou de test. La mesure de performance de la k-fold cross-validation dans son ensemble sera alors la moyenne des mesures de performance du modèle relatif à chaque pli. On peut voir à la figure 3.9 un exemple de cross validation avec $k = 5$plis.

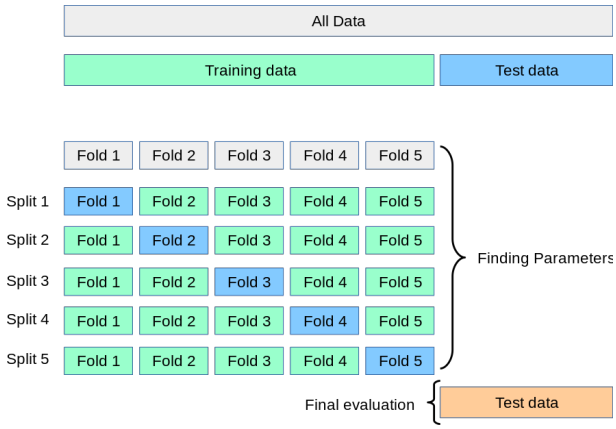


Figure 3.9 – Exemple de cross validation avec $k = 5$ plis.

Il existe plusieurs algorithmes, détaillés à la section 3.2, pouvant être utilisés pour résoudre des problèmes de classification. Pour pouvoir sélectionner le modèle le plus performant, on utilise la k-fold cross-validation sur chacun de ces algorithmes et on compare ensuite leurs performances moyennes. Après avoir identifié l'algorithme délivrant les meilleures performances pour répondre au problème de classification, il peut être utile de trouver la valeur des hyperparamètres de l'algorithme qui délivrent les meilleures performances. Chaque algorithme possède des hyperparamètres qui leur sont propres. Pour l'algorithme du SVM il s'agit par exemple de C, de gamma ainsi que du type de noyau utilisé. La valeur de ces hyperparamètres peut influencer les performances du modèle. La méthode du GridSearchCV permet d'identifier les meilleures valeurs de ces hyperparamètres. Son processus est simple. Il consiste à entraîner un modèle pour toutes les combinaisons possibles des valeurs des hyperparamètres. En plus de tester toutes les combinaisons, GridSearchCV va effectuer une k-fold cross-validation sur chacune des combinaisons. Il calcule ensuite les performances moyennes pour pouvoir ainsi identifier les valeurs des hyperparamètres qui permettent au modèle d'obtenir les meilleures performances.

Toujours dans la partie de prétraitement des données, il est important de standardiser les données. Si on ne le fait pas, les variables d'entrées avec des valeurs absolues plus grandes auront un poids plus important dans le modèle. Il faut donc standardiser les données pour qu'elles suivent toutes la loi normale centrée réduite $N(0, 1)$ avec une moyenne de 0 et une déviation standard de 1. Pour ce faire, on calcule le **Z-score** pour chaque instance. Le Z-score correspond au nombre d'écarts types qui sépare une instance de la valeur moyenne d'une variable.

$$Zscore = \frac{X - \mu}{\sigma}$$

On peut voir à la figure 3.10 un exemple de standardisation sur le dataset Iris. Après standardisation, les données sont remises à la même échelle Le dataset Iris est l'un des datasets les plus connus dans le machine learning, il fut présenté par Fisher (1936) comme exemple d'application de l'analyse discriminante linéaire pour la classification iris selon l'espèce à laquelle ils appartiennent.

	sepal length	sepal width	petal length	petal width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Standardization →

	sepal length	sepal width	petal length	petal width
0	-0.900681	1.032057	-1.341272	-1.312977
1	-1.143017	-0.124958	-1.341272	-1.312977
2	-1.385353	0.337848	-1.398138	-1.312977
3	-1.506521	0.106445	-1.284407	-1.312977
4	-1.021849	1.263460	-1.341272	-1.312977

Figure 3.10 – Dataset Iris avant et après standardisation des données.

3.4 Critères de qualités

Il existe plusieurs mesures de performance d'un modèle de machine learning. Tout d'abord, il faut comprendre ce qu'est la matrice de confusion. Il s'agit d'une matrice qui résume les résultats obtenus lors de la prédiction du modèle en les comparant avec les valeurs cibles comme on peut le voir à la figure 3.11.

Elle utilise les concepts suivants :

— **True negative (TN)** : Elément correctement prédit comme étant équivalent à la classe 0.

— **False negative (FN)** : Elément faussement prédit comme étant équivalent à la classe 0 alors qu'il valait en réalité 1.

— **True positive (TP)** : Elément correctement prédit comme étant équivalent à la classe 1.

— **False positive (FP)** : Elément faussement prédit comme étant équivalent à la classe 1 alors qu'il valait en réalité 0.

		Predicted class	
		0	1
True class	0	9 true negative (tn)	1 false positive (fp)
	1	0 false negative (fn)	1 true positive (tp)

Figure 3.11 – Exemple de matrice de confusion.

À partir de la matrice de confusion des mesures sont calculées pour évaluer la performance d'un modèle de classification. Pour les algorithmes de classification, les principales mesures de performances utilisées sont les suivants :

L'Accuracy : permet de connaître la proportion de bonnes prédictions par rapport à toutes les prédictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

La précision : indique le nombre d'instances correctement attribuées à la classe i par rapport au nombre total d'instances prédits comme appartenant à la classe i. Autrement dit, parmi les instances prédites comme appartenant à la classe i, combien sont correctement prédites ? Si l'on cherche à limiter le nombre de faux positifs, c'est cette mesure qu'on préférera utiliser.

$$Precision = \frac{TP}{TP + FP}$$

Le Recall : indique le nombre d'instances correctement attribuées à la classe i par rapport au nombre total d'instances appartenant à la classe i. Autrement dit, parmi toutes les instances de la classe i, combien ont été correctement prédites ?

$$Recall = \frac{TP}{TP + FN}$$

Le F1-score : permet, lui, d'évaluer globalement le modèle en combinant la précision et le recall. Il est moins influencé par un jeu de données qui ne serait pas correctement balancé, qui contiendrait plus de valeurs appartenant à l'une ou l'autre classe.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

La courbe ROC (Receiver Operating Characteristic) : Comme l'indique Fawcett (2004) la courbe ROC permet de visualiser et de sélectionner les modèles sur base de leurs performances en montrant graphiquement le taux de vrais positifs (TP) par rapport au nombre de faux positifs (FP).

Elle se fonde sur deux concepts complémentaires :

— **Taux de vrais positifs (TPR)** identique au rappel, il mesure la proportion des prédictions positives qui l'était réellement. Elle permet ainsi de voir le nombre de prédictions positives qui ont échapées au modèle.

$$TPR = \frac{TP}{TP + FN}$$

— **Taux de faux positifs (FPR)** à l'inverse mesure le taux de prédictions positives incorrectement prédites.

$$FPR = \frac{FP}{FP + TN}$$

Graphiquement, au plus la courbe ROC du modèle se rapproche du coin supérieur gauche, au mieux le modèle classifie correctement les données. On peut voir à la figure 3.12 que le modèle d'analyse discriminante linéaire est plus performant que le modèle de régression logistique sur base de la courbe ROC. Pour pouvoir quantifier cette performance, on utilise la mesure **AUC (Area Under the Curve)** qui, comme son nom l'indique, mesure la surface en dessous de la courbe. Plus celle-ci est proche de 1, plus les performances du modèles sont élevées.

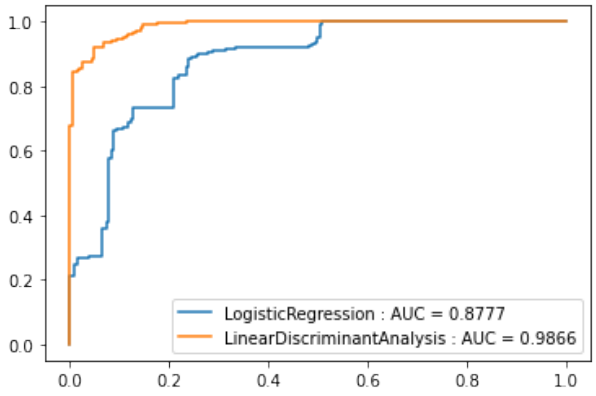


Figure 3.12 – Exemple de courbe ROC pour un modèle de régression logistique et un modèle d'analyse discriminante linéaire.

4 Analyse des données et résultats

Cette partie détaille en profondeur la collecte des données, leurs transformations et leurs nettoyages pour pouvoir être utilisées par un modèle de machine learning. Ensuite il y est expliqué les étapes permettant de sélectionner l'algorithme le plus performant. Les résultats du modèle utilisant l'algorithme le plus performant sont par la suite analysés. Enfin, le modèle est utilisé pour prédire la composition d'équipe sur une nouvelle saison.

4.1 Collecte de données

Pour pouvoir utiliser des algorithmes de classification, il faut avoir une matrice X contenant les variables d'entrées et une matrice y contenant la variable cible que l'on cherche à prédire. Ainsi, pour ce modèle, il faut que les données soient dans la structure suivante :

$$X_{p\times n} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{p1} & x_{p2} & \cdots & x_{pn} \end{bmatrix}, y_p = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

Figure 4.1 – La matrice d'entrées X et la variable cible y qui indique si un joueur a joué (1) ou non (0) lors d'un match de football où p représente le nombre de joueurs cumulés pour tous les matchs de La Liga 2020_2021 et n le nombre de variables descriptives des joueurs.

4.1.1 Données StatsBomb

Premièrement, il nous faut collecter les données que l'on cherche à prédire, à savoir la composition de l'équipe pour chaque match de championnat pour une saison entière. Plus précisément, le modèle se concentrera sur la prédiction des joueurs qui devraient jouer pendant le match. Cette information est fournie par StatsBomb dans son Open data. Les données sont fournies dans le format JSON directement depuis l'API de StatsBomb sous la structure suivante :

— Les compétitions et leurs saisons respectives sont stockées dans competition.json. On y accède par python en exécutant $sb.competitions()$. L'annexe 1 montre les données récupérées pour la saison 2020/2021 du championnat La Liga.

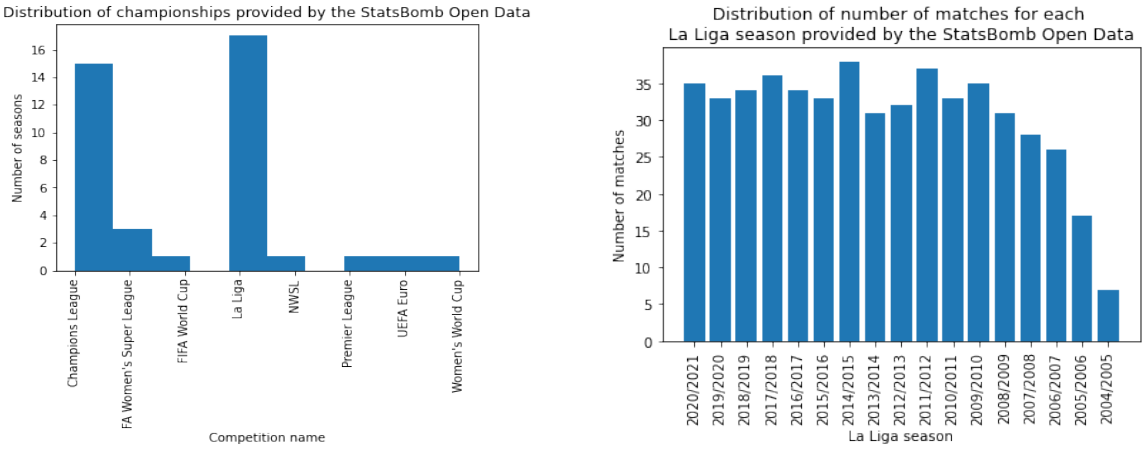
— Les données relatives à tous les matchs d'une saison pour une compétition sont stockées dans matches.json. On y accède par python en exécutant $sb.matches(competition\_id, season\_id)$. L'annexe 2 montre les données récupérées pour tous les matchs de la saison 2020/2021 du championnat La Liga.

— Les données relatives à la composition d'équipe pour chaque match sont stockées dans lineups.json. On y accède par python en exécutant $sb.lineups(match\_id)[team\_name]$. L'annexe 3 montre les données récupérées pour tous les matchs de la saison 2020/2021 du championnat La Liga.

— Les données relatives à tous les évènements (passes, tirs, fautes, ...) qui se passent pendant un match. Ces données ne seront pas utilisées pour la réalisation du modèle de cette thèse.

Ce qui nous intéresse en priorité est, ici, les données stockées dans les fichiers lineups.json de chaque match. Cela va nous permettre d'extraire la variable cible indiquant si un joueur a joué ou non pendant un match. C'est cette variable cible que le modèle de machine learning cherchera à prédire.

La figure 4.2a montre la distribution des championnats fournis par l'Open Data de StatsBomb. On se rend compte que la majorité des saisons couvertes par l'Open Data concerne la Champions League et le championnat La Liga. En creusant plus en profondeur dans les données, on s'apperçoit que pour certaines compétitions, tous les matchs ne sont pas disponibles. Par exemple pour la Champions League de la saison 2018/2019 n'est fourni que le match de la finale. Il semble donc préférable de se concentrer uniquement sur les matchs de La Liga. La figure 4.2b montre la distribution des matchs pour chaque saison disponible de La Liga. Il y a entre 30 et 36 matchs en moyenne pour les saisons entre 2008/2009 et 2020/2021. Il s'agit en fait uniquement des matchs du FC Barcelone pour chaque saison.



(a) (b)

Figure 4.2 – (a) Histogramme représentant le nombre de saisons pour chaque compétition disponible dans l'open data de StatsBomb et (b) Nombre de matchs pour chaque saison du championnat La Liga disponible dans l'open data de StatsBomb.

On décide donc de se concentrer sur les matchs du FC Barcelona puisque StatsBomb fournit les données relatives aux matchs de toute une saison pour cette équipe uniquement. Pour rester dans des données récentes, on décide arbitrairement de prendre les données StatsBomb des saisons 2017/2018, 2018/2019, 2019/2020 et 2020/2021. Les trois premières saisons serviront à entraîner le modèle et la saison 2020/2021 permettra d'évaluer les performances du modèle sur une toute nouvelle saison.

Récoltons maintenant les données des compositions d'équipes du FC Barcelone. Pour ce faire, il faut d'abord récupérer les données de tous les matchs de chacune de ces saisons. On peut voir à l'annexe A.1 les données concernant tous les matchs de la saison 2020/2021. Ayant récupéré ces données, nous pouvons, pour chaque match, grâce au match_id, accéder aux données relatives à la composition d'équipe du FC Barcelone contenues dans les fichiers lineups.json. L'annexe A.2 montre les informations de la composition d'équipe du FC Barcelone pour le match du 15/03/2021 entre Barcelone et Huesca.

L'ensemble des données stockées dans les différents fichiers lineups.json nous permet d'ex-

traire des informations telles que de savoir si un joueur a joué pendant le match ou non. Un joueur a joué pendant un match s'il a commencé le match en tant que titulaire ou s'il est monté pendant le match suite à un remplacement.

Cette information est stockée dans une variable binaire appelée *played* qui indique si oui (1) ou non (0) un joueur a joué pendant le match. On considère que tout joueur pour lequel il y a des informations concernant les positions occupées pendant un match dans le fichier lineups.json relatif au match en question a joué pendant ce match. Le modèle aura donc pour but de prédire les joueurs qui ont joué ou non dans l'équipe du FC Barcelone pour les différents matchs de la saison 2020/2021.

4.1.2 Données Fifa

Après avoir récupéré l'information que l'on cherche à prédire, il nous faut des données d'entrées pour que le modèle puisse apprendre à prédire cette variable cible. Il nous faut donc des données d'entrées qui fournissent des informations sur les joueurs. Pour ce faire, on utilise les données issues du jeu vidéo FIFA. Il s'agit en réalité de jeux de données disponibles sur Kaggle [1]. Ces données sont extraites, via du webscrapping, du site web Sofifa.com qui est le site web de référence pour trouver toutes les données des joueurs pour chaque édition du jeu FIFA. Il faut savoir que EA sport, l'éditeur du jeu, sort chaque année le jeu aux alentours de fin septembre. Ainsi, FIFA 21 sort aux alentours de fin septembre 2020. Les données de FIFA 21 correspondent donc à la saison 2020/2021.

Pour cette saison-là, Ce jeu de donnée est de taille 18944 lignes × 106 colonnes. Il contient donc 106 caractéristiques par joueur pour un total de 18944 joueurs à travers le monde. Néanmoins, uniquement les joueurs du FC Barcelone nous intéressent pour cette thèse. En sélectionnant uniquement ces derniers, il nous reste 33 joueurs. Il s'agit de tous les joueurs qui appartenaient au FC Barcelona lors du début de la saison 2020/2021. En regardant en détail le nom des joueurs présents dans ce dataset, on se rend compte qu'il y a certains joueurs qui ne devraient pas s'y trouver. Des joueurs tels que Luis Suarez, Rafinha, Nelson Semedo ont par exemple été vendus par le FC Barcelone lors de l'été 2020. On se rend compte en fait que le web scrapping a été effectué sur la toute première version de l'édition 2021 de FIFA qui sort avant la fin de la période des transferts. Le jeu est ensuite adapté via des mises à jour pour rester fidèle à la réalité. Mais les données issues du web scrapping ne l'ont pas été. On retire donc manuellement tous les joueurs qui ont été vendus ou prêtés avant le début de la saison. Il en va de même pour les joueurs qui auraient éventuellement joué un match de coupe avec l'équipe A du FC Barcelone, celle qui nous intéresse, mais qui aurait joué tout le restant de la saison avec l'équipe B. Il s'agit de joueurs tels que Ludovit Reis, Rey Manaj ou encore Juan Miranda.

Par ailleurs, on se rend compte que les données attribuées aux gardiens de buts étant fortement éloignées des données des joueurs de champs, on décide de retirer les gardiens de buts de l'ensemble de données des joueurs du FC Barcelone. Le modèle ne prédit donc pas si un gardien doit ou non jouer pour un match déterminé. Après avoir effectué ce tri, il reste donc 21 joueurs au sein de l'équipe du FC Barcelone. Le nombre de joueurs n'étant pas forcément fixe d'années en années, il y en a 20 pour la saison 2017/2018, 20 pour la saison 2018/2019 et 21 pour la saison 2019/2020.

1. `https://www.kaggle.com/datasets/stefanoleone992/fifa-22-complete-player-dataset`

4.2 Transformation des données

On doit maintenant regrouper les données issues de FIFA relatives aux caractéristiques des joueurs avec les données issues de l'Open Data de StatsBomb nous indiquant quels joueurs ont joué pour les différents matchs. On doit donc pour chaque match, récupérer les données FIFA des 23 ou 24 joueurs du FC Barcelone ainsi que les données StatsBomb concernant les informations relatives à la composition d'équipe pour le match en question. On effectue ensuite un lent merge sur `player_id` avec le dataset des données Fifa des joueurs du FC Barcelona à gauche et le dataset des données des lineups StatsBomb pour le match en question à droite. On obtient donc un dataset fusionné regroupant pour chaque match d'une saison les caractéristiques du match, celles des joueurs et l'information quant au fait qu'ils aient ou non joué pendant le match. Les caractéristiques du match sont des informations telles que l'identifiant du match, le nom de l'équipe à domicile et à l'extérieur, ainsi que les résultats du match. On va ensuite concaténer l'ensemble des datasets fusionnés de chaque match de la saison pour obtenir un seul dataset dans lequel on retrouve les 35 datasets correspondants aux 35 matchs. On obtient donc par exemple, un dataset de 735 lignes × 124 colonnes pour la saison 2020/2021. À savoir 21 joueurs x 35 matches.

Étant donné que l'on va entraîner le modèle sur les saisons 2017/2018, 2018/2019 et 2019/2020, on répète l'opération pour chaque de ces saisons et on effectue la concaténation de chacun de ces datasets. On conserve quant à lui le jeu de donnée de la saison 2020/2021 de côté pour l'évaluation du modèle. On en a maintenant fini pour ce qui est de la collecte de données.

4.3 Nettoyage des données

Maintenant que nous avons un seul et unique dataset qui contient toutes les informations intéressantes, il faut s'assurer qu'il n'y ait pas de problèmes de validité de données, de données manquantes, ou d'incohérence dans les données.

En regardant dans les données issues du dataset FIFA, on se rend compte que les gardiens ne possèdent pas du tout les mêmes données que les joueurs de champ. Il y a des variables qui ne contiennent des valeurs que pour les gardiens. Il s'agit des variables *gk_diving*, *gk_handling*, *gk_kicking*, *gk_reflexes*, *gk_speed* et *gk_positioning* qui sont des variables qui stockent les informations propres aux gardiens uniquement. Les joueurs de champs ont eux des valeurs NaN pour toutes ces variables. Les valeurs NaN (Not a Number) sont utilisées pour représenter des données manquantes et qui ne peuvent pas être utilisées dans des matrices d'entiers comme expliquées par McKinney et al. (2010). Ayant retiré les gardiens du jeu de données, il n'est plus utile de conserver ces variables dans le dataset.

Ensuite, on s'aperçoit que deux colonnes du dataset contiennent uniquement que des NaN values. Il s'agit de *loaned_from* and *defending_marking*. Ces variables ont pour but d'indiquer respectivement depuis quel club le joueur est prêté, à condition qu'il soit prêté, et la capacité de marquage défensif du joueur. Pour ce qui est de *loaned_from*, en regardant en détail dans les données du dataset Fifa, on s'aperçoit qu'elle contient des valeurs, autres que NaN. Il ne s'agit donc pas d'un problème de données mais, cela veut juste dire que le FC Barcelone n'avait aucun de leurs joueurs qui étaient prêtés par un autre club. On retire malgré tout cette variable du jeu de données puisqu'elle ne stocke aucune information. Alors que pour la variable *defending_marking*, même en regardant dans le dataset FIFA, elle ne contient que des valeurs NaN. En l'état cette variable ne nous sert donc à rien. En cherchant plus loin parmi les datasets

des autres jeux FIFA, donc dans les datasets de FIFA 16 à FIFA 22, on se rend compte qu'il s'agit d'une variable qui avait des valeurs dans les jeux de données de FIFA 16 à FIFA 19 inclus mais qui n'en a plus eu dans les FIFA suivants. En fouillant sur Sofifa.com qui est donc le site web depuis lequel les données ont été extraites, on s'aperçoit qu'il s'agit en fait d'un changement de nom de la variable. Avant FIFA 20, la variable s'appelait *Marking* et soudainement à partir de FIFA 20 la variable a été renommée *defensive_awareness* dans le jeu vidéo. Néanmoins, la variable *defensive_awareness* n'existe pas dans les datasets de FIFA 20 et des jeux postérieurs. Or, dans la logique on devrait retrouver la variable *Marking* pour les datasets jusqu'à FIFA 19 inclus et ensuite la variable devrait s'appeller *defensive_awareness*. La raison de ce problème vient probablement d'une mauvaise conception de l'algorithme de scrapping qui a été utilisé pour récupérer les données depuis Sofifa.com. L'algorithme récupère probablement le nom des variables ainsi que les valeurs associées à ces variables pour chaque joueur et chaque année d'édition du jeu. Mais il prend probablement les noms des variables à partir d'un des jeux avant le changement de nom sur FIFA 20. Ainsi, l'algorithme n'a pas pu récupérer la valeur de *Marking* pour FIFA 20 et les suivants puisque cette variable n'existait plus avec exactement le même nom. Dans l'idéal pour résoudre ce problème, il faudrait faire la modification de la valeur de *Marking* pour tous les joueurs en prenant donc la valeur de *defensive_awareness* pour les FIFA postérieurs à FIFA 20 inclus. Néanmoins, ayant déjà 105 autres variables, et la valeur de la variable *Marking* ou *defensive_awareness* étant indirectement incluses dans la variable *defending* qui résume toutes les sous-variables de défenses, nous décidons simplement de ne pas tenir compte de cette variable et de la retirer du jeu de données.

Il y a également les colonnes *nation_position* et *nation_jersey_number* qui ne contiennent, par exemple, toutes les deux seulement 245 valeurs valides sur les 735 lignes du dataset de la saison 2020/2021. Étant donné qu'il s'agit d'informations à propos des joueurs qui sont appelés par les fédérations nationales pour aller jouer en équipe nationale, seuls les joueurs appelés par le coach de l'équipe nationale ont des valeurs valides. Les autres joueurs ont, eux, des valeurs NaN. S'agissant du numéro de maillot et de la position en équipe nationale, ces variables pourraient indirectement avoir de l'importance pour prédire si un joueur doit être sélectionné ou non. Non pas que le fait de connaître le numéro de maillot d'un joueur soit important pour pouvoir prédire la sélection ou non d'un joueur pour un match déterminé en championnat. Toutefois sans pouvoir l'affirmer avec certitude. Mais ce qui nous intéresse davantage est le fait de savoir si un joueur est sélectionné dans l'équipe nationale. Il paraît évident qu'un joueur ayant de bonnes performances soit souvent appelé en équipe nationale. Nous avons donc extrait cette information en créant une nouvelle variable *selected_in_national_team* en considérant qu'un joueur ayant un *nation_position* et un *nation_jersey_number* avait déjà été appelé en équipe nationale et que les autres joueurs n'ayant pas de valeurs pour ces variables ne l'avaient donc pas été.

La variable cible *played* ne contient des valeurs que pour 629 des 735 lignes du dataset. Lorsqu'elle a été extraite de l'Open Data de StatsBomb, et qu'un joueur n'était pas sélectionné et donc pas présent dans le fichier lineups.json qui ne reprend que les joueurs sélectionnés pour un match. Ainsi, lors de la fusion entre le dataset contenant les caractéristiques FIFA de tous les joueurs du FC Barcelone et le jeu de données de la composition d'équipe pour chaque match, des valeurs NaN ont été ajoutées pour les joueurs du dataset de gauche, les caractéristiques FIFA, n'étant pas présent dans le dataset de droite, à savoir la composition d'équipe. Il suffit donc juste de remplacer toutes les valeurs NaN par 0. Pour ce faire, il suffit d'utiliser la fonction fillna() qui permet de remplacer toutes les valeurs NaN d'une variable par 0.

On vérifie également qu'il n'y ait pas de valeurs absurdes pour les variables numériques telles que des valeurs négatives qui n'auraient pas de sens. Il s'avère qu'aucune anomalie n'est détectée. Les modèles de Machine Learning ne prennent en compte uniquement les variables numériques. Ayant déjà veillé à extraire les informations utiles des variables catégorielles dans des variables numériques, nous pouvons ne garder que les variables numériques. Pour rappel, le modèle sera entraîné sur les saisons 2017/2018, 2018/2019 et 2019/2020. Ce dataset d'entraînement est de taille 2093 lignes avec 65 colonnes. La saison 2020/2021, quant à elle, sera gardée pour la validation du modèle et son dataset est de taille 735 lignes pour 65 colonnes.

4.4 Sélection du modèle

Dans cette section nous allons passer à la phase d'entraînement des modèles des différents algorithmes de classification pour pouvoir comparer leurs performances et sélectionner l'algorithme dont le modèle est le plus performant pour prédire si un joueur du Fc Barcelone doit jouer ou non pour chacun des matchs de la saison 2020/21.

Commençons par définir le vecteur cible y comme correspondant à la variable que l'on cherche à prédire, à savoir $played$. On attribue alors toutes les 64 autres variables numériques à la matrice X. On a donc X et y :

$$X_{1085\times 65} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,65} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,65} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1085,1} & x_{1085,2} & \cdots & x_{1085,65} \end{bmatrix}, y_{1085x1} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

Figure 4.3 – La matrice d'input X et la variable cible y qui indique si un joueur du FC Barcelone a joué (1) ou non (0) lors de chaque match de la saison 2020/21 de La Liga.

Il faut maintenant standardiser la matrice X pour mettre toutes les variables à la même échelle. Comme le montre l'annexe A.1 la standardisation des données permet d'améliorer les performances des modèles de classification.

Ensuite, pour déterminer l'algorithme qui permet d'obtenir les meilleures performances de classification, nous utilisons la méthode de k-fold cross-validation avec $k = 10$ plis que l'on va effectuer sur 6 algorithmes différents issus de la librairie scikit-learn : la régression logistique (LR), l'analyse discriminante linéaire (LDA), la méthode des k plus proches voisins (KNN), l'arbre de décision pour la classification, la classification naïve bayesienne gaussienne (GaussianNB) et la machine à vecteurs de supports pour la classification (SVC). Il faut noter qu'avec la méthode de k-fold cross-validation il n'est pas facile d'évaluer les performances du modèle sur un dataset laissé de côté comme l'est celui de la saison 2020/2021. Pour cette partie de sélection du modèle, seul le dataset d'entraînement sera utilisé. Il n'empêche qu'à chaque itération d'un algorithme les données seront séparées en données d'entraînement et de validation.

Pour mesurer les performances des différents algorithmes, nous utilisons les mesures détaillées à la section 3.4. Pour rappel, l'accuracy permet d'indiquer le taux de bonnes prédictions. Le recall, lui, correspond au nombre de joueurs prédits comme devant être sélectionnés par rapport au nombre de joueurs effectivement sélectionnés. La précision indique, ici, le nombre de fois où la sélection d'un joueur a été correctement prédite par rapport au nombre de sélections

prédites qu'elles soient correctes ou non. Le F1-score est une combinaison du recall et de la précision. Enfin, la mesure Aera Under the Curve (AUC) correspond à l'aire sous la courbe ROC qui, elle, permet de montrer le taux de vrais positifs par rapport au nombre de faux positifs. Plus la valeur de chacune de ces mesures est proche de 1, meilleures sont les performances du modèle. Comme on peut le voir à la table 4.1, les performances moyennes des modèles sont en général relativement bonnes avec des scores compris entre 69% et 81% pour la majorité des mesures de performances pour les différents modèles. On remarque cependant que les performances de l'algorithme de classification naïve bayésienne et de l'algorithme des arbres de décisions sont un peu moins bonnes en moyenne que les autres algorithmes.

Comparaison performances moyennes des algorithmes de classification sklearn					
	Accuracy	Recall	Precision	F1-score	ROC-AUC
LR	0.7583	0.7972	0.7929	0.7936	0.8167
LDA	0.7588	0.8111	0.7873	0.7982	0.8121
KNN	0.7493	0.8069	0.7772	0.7909	0.7973
DecisionTree	0.7091	0.7496	0.7547	0.7500	0.7043
GaussianNB	0.6905	0.7540	0.7303	0.74122	0.7776
SVC	0.7650	0.7919	0.8059	0.7975	0.8137

Table 4.1 – Moyenne des performances obtenues pour chaque modèle avec une k-fold cross-validation de 10 folds en utilisant model_selection.cross_val_score(). LR = Régression logistique, LDA = Analyse discriminante linéaire, KNN = K plus proches voisins, SVM = Machine à vecteurs de supports.

Les autres algorithmes ont des performances assez proches les uns des autres. Néanmoins, regardons les diagrammes de boîte à moustache des différents algorithmes pour les différentes mesures de performances. Comme on peut le voir à l'annexe A.2, l'analyse discriminante linéaire et la machine à vecteurs de supports(SVM) sont les deux algorithmes avec les meilleures performances en termes d'accuracy. Pour rappel, il s'agit d'une mesure permettant de calculer le taux de bonnes prédictions. La régression logistique a aussi un bon score d'accuracy mais la distance entre le troisième quartile et le maximum est étonnamment très petite.

Le recall correspond au nombre de joueurs prédits comme devant être sélectionnés par rapport au nombre de joueurs effectivement sélectionnés. L'annexe A.3 montre que, les performances de l'analyse discriminante linéaire semblent meilleures que celles de la régression logistique avec une moyenne de 0.81. L'algorithme des k plus proches voisins a lui aussi une moyenne de 0.81 mais possède des résultats qui sont moins dispersés que l'algorithme d'analyse discriminante linéaire. À l'inverse, les résultats du SVM sont, eux, très dispersés.

La précision indique, ici, le nombre de fois où la sélection d'un joueur a été correctement prédite par rapport au nombre de sélections prédites qu'elles soient correctes ou non. L'annexe A.4 montre que la régression logistique a des bonnes performances mais qu'elle possède un outlier, représenté par le point. Un des modèles lors de la k-fold cross-validation a donc eu un score de précision d'environ 87% qui est considéré comme anormal. L'analyse discriminante linéaire et le SVM ont, eux, des performances assez similaires. Ils n'ont pas d'outliers mais ont malgré tout un maximum assez élevé qui s'approche de la valeur de l'outlier de la régression logistique. L'algorithme des KNN a eu 4 valeurs considérées comme anormales. L'algorithme de classification naïve bayésienne et celui des arbres de décision conservent des performances légèrement inférieures aux autres algorithmes.

Concernant le F1-score, qui est une combinaison du recall et de la précision, l'annexe A.5

montre que la régression logistique, l'analyse discriminante linéaire et le SVM ont des performances très similaires avec toutefois une médiane davantage décentrée pour le SVM. L'algorithme KNN est le seul à disposer d'un outlier.

Enfin, en regardant les performances des modèles en termes de surface sous la courbe ROC, l'annexe A.6 montre que l'algorithme avec la meilleure performance AUC est le SVM, suivi de près par la régression logistique, qui a néanmoins une médiane davantage décentrée, et le KNN. L'analyse discriminante linéaire a un résultat qualifié d'anormal qui est assez faible en termes de performances AUC. L'algorithme de classification naïve bayésienne et celui des arbres de décision conservent des performances légèrement inférieures aux autres algorithmes.

Ainsi si l'on doit sélectionner un des six algorithmes pour notre modèle, il est tout d'abord clair que l'algorithme de classification naïve bayésienne et celui des arbres de décisions semblent avoir des performances moindres que les quatre autres. L'algorithme des k plus proches voisins semble avoir des résultats assez épars avec la présence d'une valeur anormale pour le F1-score et quatre pour la précision. Pour ce qui est des trois autres algorithmes, ils ont des performances assez similaires. Néanmoins, le SVM semble, pour toutes les mesures, excepté le recall, légèrement meilleur que l'analyse discriminante linéaire qui elle paraît être légèrement meilleur que l'algorithme de régression logistique.

4.5 Analyse des résultats du modèle de machine à vecteurs de supports (SVM)

Maintenant que la machine à vecteurs de supports (SVM) est l'algorithme choisi pour notre modèle, nous pouvons optimiser les performances de ce modèle en choisissant les meilleurs hyperparamètres. GridSearch permet très rapidement de trouver les meilleurs hyperparamètres C et gamma du SVM, ainsi que de déterminer quel est le meilleur noyau. GridSearch effectue donc une cross-vallidation de 5 plis pour toutes les possibilités de modèle en combinant les différentes valeurs de C, gamma et du noyau. Pour ce modèle, GridSearch a testé des modèles avec des valeurs de C de 0.1, 1, 10, 100, 1000. Les valeurs de gamma testées équivalentes à 1, 0.1, 0.001, 0.0001 et 0.00001. Il testera ces 25 possibilités de valeurs pour les hyperparamètres sur des SVM utilisant un noyau de fonction à base radiale (RBF), un noyau linéaire, un noyau polynomial et un noyau sigmoïde. Ce qui donne un total de 100 possibilités pour lesquelles une cross-validation de 5 plis est effectuée. GridSearch va donc entraîner 500 modèles et déterminer les meilleurs hyperparamètres ainsi que le meilleur noyau sur base de la moyenne des performances de précision, qui est la mesure de performance utilisée par défaut.

De ces 100 combinaisons possibles, GridSearchcv a identifié les meilleurs hyperparamètres comme étant C = 1000, gamma = 0.001 et avec un noyau de fonction à base radiale (RBF).

Une fois les meilleurs hyperparamètres identifiés, le modèle peut encore être optimisé. On peut identifier le nombre de variables utilisées par le modèle pour optimiser ces performances. Pour ce faire, il suffit d'entraîner et d'évaluer les performances d'un modèle en prenant lors de la première itération une seule variable et en ajoutant à chaque itération une variable supplémentaire. Les variables sont ajoutées dans l'ordre décroissant de leur corrélation avec la variable cible y. La figure 4.4 montre que la variable la plus corrélée avec le fait qu'un joueur du FC Barcelone ait joué ou non pour les différents matchs des saisons 2017/2018, 2018/2019 et 2019/2020 est la variable $overall$ qui correspond à la note générale du niveau d'un joueur. Il s'agit en quelque sorte d'une moyenne de toutes les autres variables. Allant de 0 à 100, plus la note overall d'un joueur est proche de 100, plus le joueur est supposé être fort dans le jeu. Il est

donc logique que cette variable soit la plus corrélée avec la variable cible $played$. Il est également intéressant de noter que des variables telles que $wage$ qui indique le salaire du joueur, age qui indique l'âge du joueur, $value\_eur$ qui indique la valeur marchande du joueur, $release\_clause$ qui indique le montant de la clause libératoire et finalement le $player\_id$ qui est l'identifiant unique du joueur tel qu'attribué par StatsBomb sont parmi les variables les plus corrélées avec la variable $played$. Toutes ces variables contiennent des informations réelles et "externes" au jeu et qui pourraient être récupérées sans utiliser le jeu de données issues du jeu vidéo FIFA.

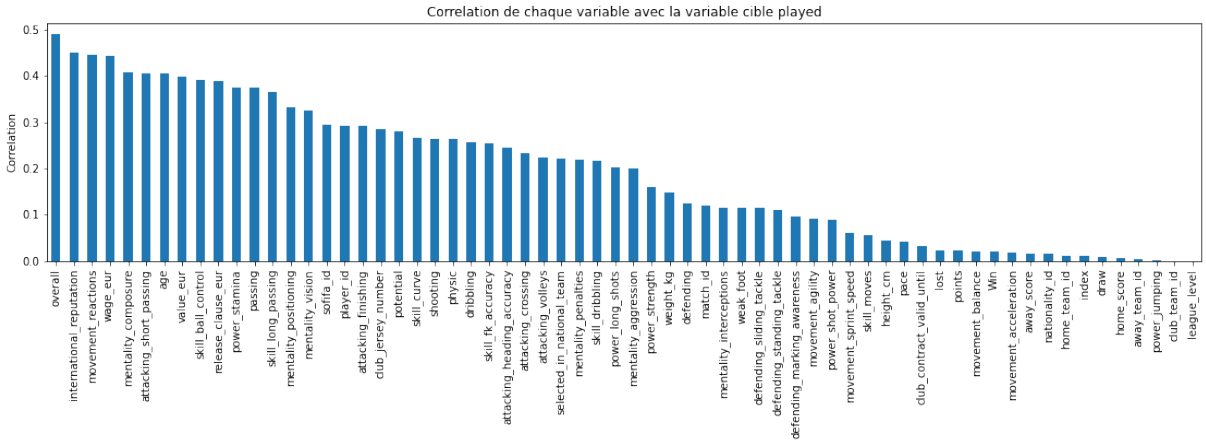
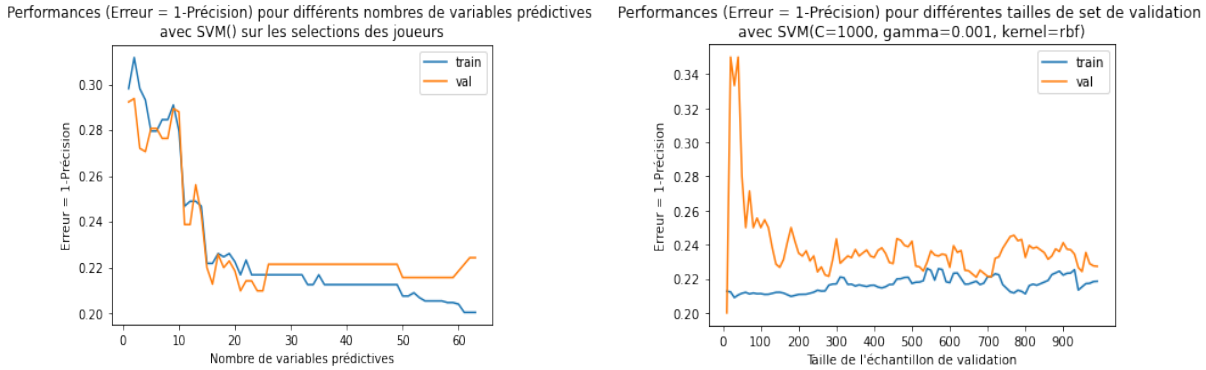


Figure 4.4 – Histogramme de la corrélation de chaque variable avec la variable played.

La figure 4.5a montre l'évolution des performances des modèles de classification $SVM(C = 1000, \gamma = 0.001, kernel = "rbf")$ en fonction du nombre de variables utilisées. Le nombre optimal de variables pour la matrice X est donc de 21 variables. La figure 4.5b montre que le modèle obtient les meilleures performances en validation lorsque la taille de l'échantillon de validation est de 670. Étant calculé sur base d'un jeu de données qui contient 2093 lignes, cela correspond à une répartition optimale de validation/entraînement équivalente à 32/68%. On peut donc maintenant évaluer les performances du modèle $SVM(C = 1000, \gamma = 0.001, kernel = "rbf")$ en prenant dans la matrice X les 21 variables les plus corrélées avec $played$ et en utilisant une répartition de validation/entraînement de 32%/68%.



(a) (b)

Figure 4.5 – (a) Graphique d'évolution des performances en fonction du nombre de variables (b) Graphique d'évolution des performances en fonction du nombre de données de validations. (a) et (b) utilisant le modèle $SVM(C = 1000, \gamma = 0.001, kernel = "rbf")$.

La figure 4.6 montre la matrice de confusion obtenue sur les données de validation par le modèle de classification. C'est sur base de cette matrice que les mesures de performances sont calculées. Ainsi, la précision pour la classe 0 se calcule comme étant $\frac{TrueNeg}{TrueNeg+FalseNeg}$, soit $\frac{199}{199+57} = 0.78$. À l'inverse, le recall pour la classe 0 se calcule comme étant $\frac{TrueNeg}{TrueNeg+FalsePos}$, soit $\frac{199}{199+84} = 0.70$. La table 4.2 montre donc toutes les mesures de performances. On peut y voir un score de précision similaire pour la classification de la classe 1 (a joué) et 0 (n'a pas joué) avec un score de 0.78 et de 0.80. Il y a par contre une grande différence dans la performance du modèle en termes de recall pour la classe 1 et la classe 0. Cela veut donc dire que le modèle est moins bon pour prédire le nombre de joueurs devant ne pas jouer que pour prédire le nombre de joueurs devant être sélectionnés.

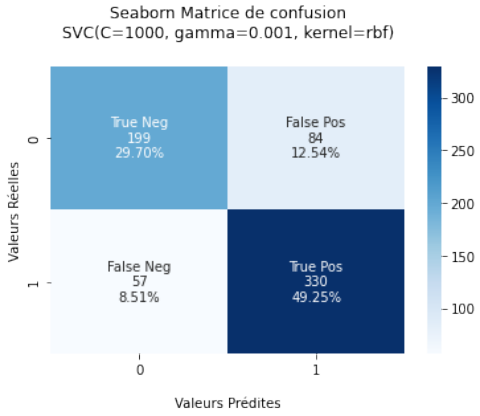


Figure 4.6 – Matrice de confusion obtenue grâce au modèle $SVM(C = 1000, \gamma = 0.001, kernel = "rbf")$ sur les saisons 2017/2018, 2018/2019, 2019/2020 de La Liga.

Performance de classification du $SVM(C = 1000, \gamma = 0.001, kernel = "rbf")$				
	Precision	Recall	F1-score	support
0.0	0.78	0.70	0.74	283
1.0	0.80	0.85	0.82	387
accuracy			0.79	670
macro avg	0.79	0.78	0.78	670
weighted avg	0.79	0.79	0.79	670

Table 4.2 – Performances de classification de la sélection (1.0) ou non (0.0) des joueurs du FC Barcelone pour la saison 2020/21 de La Liga obtenues avec le modèle de régression logistique.

4.6 Analyse des résultats sur la saison 2020/2021

Regardons maintenant les performances du modèle lorsqu'on lui demande de prédire si un joueur devrait ou non jouer sur une toute nouvelle saison qui n'était donc pas présente dans le jeu de données d'entraînement.

Pour ce faire, nous conservons le même modèle, que l'on entraîne toujours sur les trois saisons précédentes, à savoir 2017/2018, 2018/2019 et 2019/2020. Mais au lieu de séparer une partie de ces données dans un dataset d'entraînement et de validation, on utilise toutes les données de ces trois saisons pour l'entraînement du modèle et la saison 2020/2021 sera, elle, utilisée comme dataset de validation.

Comme on peut voir à la table 4.3, les performances du modèle sont nettement moins bonnes. Avec une perte d'environ 15 à 20% pour toutes les mesures de performances. Les performances de classification des joueurs ne jouant pas, s'approchent d'un modèle purement aléatoire.

La figure 4.7 montre la matrice de confusion obtenue sur les données de validation par le modèle de classification montre un taux de faux négatifs et de faux positifs assez important.

Performance de classification du $SVM(C = 1000, \gamma = 0.001, kernel = "rbf")$				
	Precision	Recall	F1-score	support
0.0	0.56	0.58	0.57	304
1.0	0.70	0.68	0.69	431
accuracy			0.64	735
macro avg	0.63	0.63	0.63	735
weighted avg	0.64	0.64	0.64	735

Table 4.3 – Performances de classification des joueurs du FC Barcelone selon les classes a joué(1.0) ou non(0.0) pour la saison 2020/21 de La Liga obtenues avec le modèle $SVM(C = 1000, \gamma = 0.001, kernel = "rbf")$.

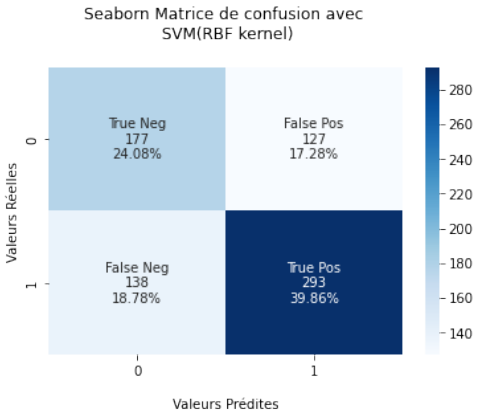


Figure 4.7 – Matrice de confusion obtenue grâce au modèle $SVM(C = 1000, \gamma = 0.001, kernel = "rbf")$ sur les données de validation de la saison 2020/2021 de La Liga.

La table 4.4 montre les résultats de classification pour le match entre Barcelone et Huesca du 15/03/2021. Tout d'abord, on peut voir qu'il y a cinq faux positifs : Piqué, M.Pjanic, Coutinho, Sergi Roberto et Ansu Fati. Tous ces joueurs ont été classifiés par le modèle comme devant jouer mais ne l'ont pas été. En cherchant des informations relatives aux périodes de blessure des

joueurs du FC Barcelone sur FootMercato.com, on se rend compte en fait que Piqué souffrait d'une blessure au genou (indisponibilité du 04/03/2021 au 09/04/2021, soit 36 jours). M.Pjanic revenait d'une blessure au pied du 25/02/2021 au 04/03/2021. Toutefois, rien n'indique qu'il était encore blessé pour ce match. Coutinho souffrait d'une blessure au genou (indisponibilité du 30/12/2020 au 30/06/2021, soit 182 jours). Sergi Roberto souffrait d'une blessure aux Ischio-jambiers (indisponibilité du 04/02/2021 au 09/04/2021, soit 64 jours). Ansu fati souffrait d'une blessure au genou (indisponibilité du 08/11/2020 au 25/09/2021, soit 321 jours). Il s'agit donc pour la plupart des faux positifs de joueurs qui souffraient de blessures importantes et qui en temps normal auraient probablement joué pendant le match.

	match_date	home_team	home_score	away_score	away_team	short_name	played	Prediction	Prediction_proba	Nb Match played in 2020/21	Nb Match played in 2019/20
0	2021-03-15	Barcelona	4	1	Huesca	L. Messi	1.0	1.0	0.9686	35.0	33.0
2	2021-03-15	Barcelona	4	1	Huesca	Sergio Busquets	1.0	1.0	0.7041	33.0	29.0
3	2021-03-15	Barcelona	4	1	Huesca	A. Griezmann	1.0	1.0	0.8715	33.0	30.0
4	2021-03-15	Barcelona	4	1	Huesca	Piqué	0.0	1.0	0.6388	18.0	30.0
5	2021-03-15	Barcelona	4	1	Huesca	Jordi Alba	1.0	1.0	0.8458	33.0	23.0
6	2021-03-15	Barcelona	4	1	Huesca	M. Pjanić	0.0	1.0	0.7339	17.0	Not in the team
7	2021-03-15	Barcelona	4	1	Huesca	C. Lenglet	1.0	0.0	0.3992	32.0	23.0
8	2021-03-15	Barcelona	4	1	Huesca	F. de Jong	1.0	1.0	0.8843	34.0	24.0
9	2021-03-15	Barcelona	4	1	Huesca	Coutinho	0.0	1.0	0.6592	11.0	Not in the team
10	2021-03-15	Barcelona	4	1	Huesca	Sergi Roberto	0.0	1.0	0.7996	15.0	26.0
11	2021-03-15	Barcelona	4	1	Huesca	S. Umtiti	0.0	0.0	0.0899	11.0	13.0
12	2021-03-15	Barcelona	4	1	Huesca	O. Dembélé	1.0	0.0	0.2309	27.0	4.0
14	2021-03-15	Barcelona	4	1	Huesca	Junior Firpo	0.0	0.0	0.2705	5.0	15.0
15	2021-03-15	Barcelona	4	1	Huesca	Trincão	1.0	0.0	0.0711	25.0	Not in the team
16	2021-03-15	Barcelona	4	1	Huesca	M. Braithwaite	1.0	0.0	0.2744	26.0	Not in the team
17	2021-03-15	Barcelona	4	1	Huesca	Ansu Fati	0.0	1.0	0.7066	7.0	Not in the team
18	2021-03-15	Barcelona	4	1	Huesca	Riqui Puig	1.0	1.0	0.7845	11.0	11.0
19	2021-03-15	Barcelona	4	1	Huesca	Aleñá	0.0	0.0	0.1787	2.0	3.0
20	2021-03-15	Barcelona	4	1	Huesca	Pedri	1.0	1.0	0.6379	35.0	Not in the team
21	2021-03-15	Barcelona	4	1	Huesca	Matheus Fernandes	0.0	0.0	0.4698	0.0	Not in the team
22	2021-03-15	Barcelona	4	1	Huesca	R. Araujo	1.0	0.0	0.3176	21.0	Not in the team

Table 4.4 – Résultats de classification obtenus par le modèle de régression logistique sur les joueurs du FC Barcelone pour le match du 25/04/2021 entre Villarreal et le FC Barcelone.

Concernant les faux négatifs, on peut recenser C.Lenglet, O.Dembélé, Trincao, M.Braithwaithe et R.Araujo. Premièrement, au vu des absents à cause des blessures, l'entraîneur a dû choisir d'autres joueurs pour les remplacer. Or, ces joueurs n'auraient peut-être pas joué en temps normal. En effet, C.Lenglet et Trincao ont, par exemple, été prêtés dans d'autres clubs lors de la période du mercato de l'été 2022. Cependant, pour O.Dembélé, on voit qu'il n'a joué que quatre matchs lors de la saison précédente (2019/2020). En se basant donc sur le nombre de matchs joués des saisons précédentes, le modèle a estimé qu'il ne devrait pas jouer. En fait, il n'a pratiquement pas joué lors de la saison 2019/2020 à cause d'une déchirure musculaire de la cuisse (indisponibilité du 28/11/2019 au 11/08/2020, soit 257 jours). Les blessures semblent donc jouer un rôle important dans le fait qu'un joueur joue ou non pendant un match. Or, cette information n'est pas incluse dans le modèle par manque de donnée sur les blessures des joueurs.

Le modèle arrive néanmoins, et cela est assez surprenant, à correctement classifier Pedri comme devant jouer. Or, il s'agit d'un jeune joueur de 19 ans qui sortait tout juste du centre de formation. Certes il est déjà considéré comme un très bon joueur actuellement et comme un futur grand mais étant donné qu'il n'avait jamais joué dans l'équipe A du FC Barcelone avant la saison 2020/2021, il est étonnant que le modèle arrive à le classifier correctement. Il en va de même pour Ansu Fati. Il s'agit d'une jeune pépite prometteuse qui a cependant été blessé pendant longtemps lors de la saison 2020/2021, expliquant son faible temps de jeu.

La table 4.5 montre que les joueurs ayant le plus de faux positifs ont été longtemps absents pour cause de blessures pendant la saison 2020/2021. On remarque également qu'un joueur comme Riqui Puig n'a jamais été blessé et a pourtant 24 faux positifs. Le modèle estime donc qu'il aurait dû jouer plus souvent. Riqui Puig a un profil un peu similaire à Pedri et Ansu Fati.

Il s'agit d'une jeune pépite sortie du centre de formation du FC Barcelone et qui était promis à devenir un grand joueur. Néanmoins, au début de la saison 2020/2021, Ronald Koeman, fraîchement arrivé le 19 août 2020, indique qu'il ne compte pas sur lui pour la saison à venir et lui conseille de partir en prêt. Il restera au FC Barcelone et jouera 11 matchs de championnat cette année-là. Il s'agit d'une situation relativement exceptionnelle qui est donc difficile à prévoir pour le modèle. Il est intéressant de regarder les probabilités de prédictions fournies par le modèle, comme on peut le voir à la table 4.4, et non uniquement le résultat binaire de la prédiction. On y voit par exemple qu'un joueur comme L.Messi a une probabilité de classification comme devant jouer de 0.9686 pour le match entre Barcelone et Huesca. Alors qu'un joueur comme Sergio Busquets dispose d'une probabilité plus faible valant 0.7041. On pourrait donc en déduire que si un choix doit être effectué entre ces deux joueurs, il serait conseillé par le modèle de faire jouer L.Messi plutôt que Sergio Busquets. L'annexe A.4 montre que le modèle prédit pour 11 joueurs qu'ils doivent jouer les 35 matchs de la saison, Pedri, lui, est classifié comme devant en jouer 34. Cela nous fait donc 12 joueurs considérés par le modèle comme étant les joueurs qui devraient jouer dans le plus grand nombre de matchs. Étant donné que le modèle ne prend pas en compte les gardiens, cela monte à 13 joueurs. Trois joueurs devraient eux, quand même jouer certains matchs. Alors qu'il y a six joueurs qui, selon le modèle ne devraient jamais jouer. On voit également qu'en pratique, il y a moins de joueurs qui jouent tous les matchs et également moins de joueurs qui ne jouent jamais.

Nom	Faux Positifs	Blessures 2020/21 (jours manqués)
Ansu Fati	28	321
Coutinho	24	206
Riqui Puig	24	0
Sergi Roberto	20	148
M. Pjanić	18	11
Piqué	17	121
Matheus Fernandes	8	29
Sergio Busquets	2	13
Jordi Alba	2	17
A. Griezmann	2	0
M. Braithwaite	1	33
F. de Jong	1	0

Table 4.5 – Nombre de faux positifs obtenus par le modèle pour les joueurs pour lesquels il y a eu des faux positifs.

Nom	Faux Négatifs
C. Lenglet	32
O. Dembélé	25
Trincão	25
R. Araujo	21
M. Braithwaite	19
S. Umtiti	11
Junior Firpo	5
Aleñá	2
Pedri	1

Table 4.6 – Nombre de faux négatifs obtenus par le modèle pour les joueurs pour lesquels il y a eu des faux négatifs.

4.7 Comparaison des résultats pour la sélection et la titularisation des joueurs

Jusqu'à présent, cette thèse s'est concentrée sur la classification des joueurs selon qu'ils aient ou non joué pendant un match. C'est un choix arbitraire. Mais au lieu de cela, la variable cible aurait pu être le fait qu'un joueur soit ou non sélectionné pour un match ou encore le fait qu'il soit titulaire ou non pour ce match.

La table 4.7 montre que les performances obtenues par le modèle pour prédire la sélection ou non des joueurs pour un match sont assez bonnes pour prédire la sélection (1) mais très mauvaises pour prédire la non-sélection (0). On voit que la distribution des données n'est pas correctement équilibrée. Seulement 167 données appartiennent à la classe 0 contre 568 pour la classe 1. Peut-être que les mauvaises performances sont dues à un manque de données de validation. Pour rappel, la figure 4.5b nous montre que la taille optimale de l'échantillon de validation lors de l'entraînement du modèle sur les saisons 2017/2018, 2018/2019 et 2019/2020 était de 670. Or, nous avons, certes, 735 données de validation, mais seulement 167 d'entre elles sont assignées à la classe non sélectionné (0).

Performance de classification du $SVM(C = 1000, \gamma = 0.001, kernel = "rbf")$				
	Precision	Recall	F1-score	support
0.0	0.23	0.25	0.24	167
1.0	0.78	0.76	0.77	568
accuracy			0.65	735
macro avg	0.50	0.50	0.50	735
weighted avg	0.65	0.65	0.65	735

Table 4.7 – Performances de classification des joueurs du FC Barcelone selon les classes a été sélectionné(1.0) ou non(0.0) pour un match lors de la saison 2020/21 de La Liga obtenues avec le modèle $SVM(C = 1000, \gamma = 0.001, kernel = "rbf")$.

La table 4.8 montre que les performances obtenues par le modèle pour prédire la titularisation ou non des joueurs pour un match sont plutôt mitigées. La précision est assez bonne pour la non-titularisation mais le recall est lui moins bon. À l'inverse, le recall est assez bon pour la titularisation mais la précision est, elle, moins élevée.

Performance de classification du $SVM(C = 1000, \gamma = 0.001, kernel = "rbf")$				
	Precision	Recall	F1-score	support
0.0	0.75	0.61	0.68	429
1.0	0.57	0.72	0.63	306
accuracy			0.66	735
macro avg	0.66	0.66	0.65	735
weighted avg	0.68	0.66	0.66	735

Table 4.8 – Performances de classification des joueurs du FC Barcelone selon les classes a été titularisé(1.0) ou non(0.0) pour un match lors de la saison 2020/21 de La Liga obtenues avec le modèle $SVM(C = 1000, \gamma = 0.001, kernel = "rbf")$.

5 Discussion

Le modèle de classification créé au cours de cette thèse permet de classifier les joueurs selon qu'ils doivent ou non jouer lors d'un match de championnat. Un tel modèle est, au regard de la littérature existante, une première. Les résultats obtenus par ce modèle ne sont pas encore optimaux mais restent tout de même encourageants.

Les résultats obtenus sur les saisons 2017/2018, 2018/2019 et 2019/2020 montre des performances de validation similaires aux performances lors de l'entraînement du modèle indiquant que celui-ci est capable de prédire les joueurs devant jouer ou non lorsqu'il y a déjà eu quelques matchs qui se sont déroulés pendant la saison sur laquelle le modèle est utilisé. Les erreurs de prédictions sur la saison 2020/2021 sont pour la plupart dues à des situations imprévisibles pour le modèle en l'état. Il semble évident qu'incorporer les informations des blessures et de l'indisponibilité des joueurs augmenterait les performances. Malgré des performances à première vue relativement peu optimales, en regardant en détail les prédictions fournies, on s'aperçoit que les résultats sont meilleurs qu'initialement pensés. Le modèle semble, par ailleurs, assez bon pour classifier des jeunes joueurs prometteurs comme devant jouer la plupart des matchs.

Pour augmenter les performances du modèle il faudrait incorporer plus de données. Par exemple, en augmentant le nombre de saisons et en intégrant d'autres équipes que le FC Barcelone. De plus, au lieu d'utiliser les données issues du jeu vidéo FIFA, il serait intéressant de n'utiliser que des données réelles. Pour ce faire, on pourrait utiliser les données relatives aux évènements qui se produisent pendant chaque match. On pourrait créer des mesures qui permettraient de comparer les performances réalisées dans les différents matchs par chaque joueur. Des mesures relatives aux performances sur les sept derniers jours ou sur une période plus longue pourraient également être incorporées au modèle. Il serait intéressant de voir si les performances récentes jouent un rôle ou non dans la composition de l'équipe.

Une limite importante de ce modèle réside dans sa nature. Les modèles de machine learning supervisés agissent un peu comme une boîte noire. Ils fournissent des prédictions dont la précision est évaluée à postériori au regard des évènements réels. Il est cependant difficile de savoir ce qui justifie la classification d'un joueur soit comme devant jouer pour un match mais ne devant pas jouer pour le match suivant. Il est donc difficile pour un entraîneur ou ses assistants d'avoir confiance en ce modèle. Seule la bonne performance du modèle permet de développer cette confiance. Une étude qualitative analysant l'avis de différents entraîneurs, assistants, et experts du football vis-à-vis du modèle développé dans cette thèse pourrait également être intéressant pour identifier les forces et faiblesses de ce modèle dans son application concrète.

Il est important de préciser qu'un tel modèle ne doit pas être utilisé en ne se fiant uniquement aux résultats fournis par celui-ci. Il doit être une source supplémentaire d'information lors du choix de la composition d'équipe. Il s'agit plus de dire ce qui devrait être fait, sous forme de conseils, plutôt que de prédire réellement ce qui va être fait. La prédiction sous forme de probabilité offre d'ailleurs davantage d'informations que le résultat binaire de cette prédiction.

6 Conclusion

Ce mémoire fourni la méthodologie utilisée pour la création d'un modèle de classification permettant de prédire la composition d'une équipe de football pour un match de championnat. Comme démontré dans la section 4, l'algorithme qui offre les meilleures performances pour répondre au problème de cette thèse est l'algorithme des machines à vecteurs de supports. Cependant, les performances des algorithmes de régression logistique et d'analyse discriminante linéaire étaient assez similaires. Pour ce qui est des variables ayant le plus d'impact pour la prédiction de la composition d'équipe, seule la corrélation peut, dans ce cas, fournir un début de réponse. Il est intéressant de voir que certaines variables fortement corrélées avec le fait qu'un joueur ait joué ou non pendant un match sont des variables qui ne sont pas propres au jeu vidéo FIFA. Cela donne bon espoir pour que des mesures calculées à partir de données réelles puissent être utilisées dans ce modèle.

Des travaux futurs pourraient adapter ce genre de modèle à des sélections nationales. Un club de football ayant davantage de rencontres pendant une saison de championnat qu'une sélection nationale pendant l'année, la méthodologie devrait peut-être être adaptée pour obtenir un modèle performant.

Adapter la méthodologie développée par Young and Weckman (2020) au football pourrait être très intéressant, tant elle semble fournir une source d'informations utile pour le recrutement des joueurs dans un club. Dans le même esprit que celui développé par cette thèse, de futures recherches pourraient se concentrer sur les dispositifs tactiques d'une équipe. Il s'agit de la disposition des joueurs sur le terrain. On pourrait par exemple prédire le dispositif à utiliser pour chaque match.

Enfin, ce que ne permet pas de faire le modèle actuellement est de conseiller la composition d'équipe dans le but d'optimiser les résultats de l'équipe. Ainsi, bien que le modèle intègre l'information quant à la victoire, ou non d'un match, son objectif n'est pas de maximiser les points obtenus par l'équipe. Des travaux futurs pourraient donc essayer d'inclure cela dans un modèle de classification.

Références

Baboota, R. and Kaur, H. (2018). Predictive analysis and modelling football results using machine learning approach for english premier league. International Journal of Forecasting, 35.

Brooks, J., Kerr, M., and Guttag, J. (2016). Developing a data-driven player ranking in soccer using predictive model weights. pages 49–55.

Constantinou, A. and Fenton, N. (2013). Determining the level of ability of football teams by dynamic ratings based on the relative discrepancies in scores between adversaries. Journal of Quantitative Analysis in Sports, 9 :37–50.

Duch, J., Waitzman, J., and Amaral, L. (2010). Quantifying the performance of individual players in a team activity. PloS one, 5 :e10937.

Fawcett, T. (2004). Roc graphs : Notes and practical considerations for researchers. Machine learning, 31(1) :1–38.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. Annals of eugenics, 7(2) :179–188.

Goes, F. R., Kempe, M., van Norel, J., and Lemmink, K. A. P. M. (2021). Modelling team performance in soccer using tactical features derived from position tracking data. IMA Journal of Management Mathematics, 32(4) :519–533.

Grund, T. U. (2012). Network structure and team performance : The case of english premier league soccer teams. Social Networks, 34(4) :682–690.

Gudmundsson, J. and Horton, M. (2016). Spatio-temporal analysis of team sports – a survey.

Hvattum, L. M. and Arntzen, H. (2010). Using elo ratings for match result prediction in association football. International Journal of Forecasting, 26 :460–470.

James, B. (2003). The historical baseball abstract, 4 e éd.

McKinney, W. et al. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference, volume 445, pages 51–56. Austin, TX.

Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., and Yu, B. (2019). Definitions, methods, and applications in interpretable machine learning. Proceedings of the National Academy of Sciences, 116(44) :22071–22080.

Pappalardo, L., Cintia, P., Ferragina, P., Massucco, E., Pedreschi, D., and Giannotti, F. (2019). Playerank : Data-driven performance evaluation and player ranking in soccer via a machine learning approach. ACM Transactions on Intelligent Systems and Technology, 10 :1–27.

Passos, P., Davids, K., Araujo, D., Paz, N., Minguéns, J., and Mendes, J. F. (2010). Network as a novel tool for studying team ball sports as complex social system. Journal of science and medicine in sport / Sports Medicine Australia, 14 :170–6.

Payyappalli, V. and Zhuang, J. (2019). A data-driven integer programming model for soccer clubs decision making on player transfers. Environment Systems and Decisions, 39.

Pollard, R. (2002). Charles reep (1904-2002) : pioneer of notational and performance analysis in football. Journal of Sports Sciences, 20 :853–855.

Reep, C., Pollard, R., and Benjamin, B. (1971). Skill and chance in ball games. Journal of the Royal Statistical Society. Series A (General), 134 :623.

Robberechts, P., Haaren, J. V., and Davis, J. (2021). A bayesian approach to in-game win probability in soccer. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery &amp Data Mining. ACM.

Saritas, M. M. and Yasar, A. (2019). Performance analysis of ann and naive bayes classification algorithm for data classification. International Journal of Intelligent Systems and Applications in Engineering, 7(2) :88–91.

Stöckl, M., Seidl, T., Marley, D., and Power, P. (2021). Making offensive play predictable -using a graph convolutional network to understand defensive performance in soccer.

Young, W. and Weckman, G. (2020). A team-compatibility decision support system for the national football league. International Journal of Computer Science in Sport, 19 :60–101.

A Annexes


```
{
  "competition_id" : 11,
  "season_id" : 90,
  "country_name" : "Spain",
  "competition_name" : "La Liga",
  "competition_gender" : "male",
  "competition_youth" : false,
  "competition_international" : false,
  "season_name" : "2020/2021",
  "match_updated" : "2022-02-11T14:56:09.076",
  "match_updated_360" : "2021-10-30T04:19:36.116600",
  "match_available_360" : "2021-09-17T15:18:33.787790",
  "match_available" : "2022-02-11T14:56:09.076"
}
```


Listing 1 – Exemple des données issues de competition.Json pour la saison 2020/2021 du championnat espagnol La Liga. Données récupérées via sb.competitions() où competition_id = 11 et season_id = 4.


```
 {
  "match_id" : 3773369,
  "match_date" : "2021-03-15",
  "kick_off" : "21:00:00.000",
  "competition" : {
    "competition_id" : 11,
    "country_name" : "Spain",
    "competition_name" : "La Liga"
  },
  "season" : {
    "season_id" : 90,
    "season_name" : "2020/2021"
  },
  "home_team" : {
    "home_team_id" : 217,
    "home_team_name" : "Barcelona",
    "home_team_gender" : "male",
    "home_team_group" : null,
    "country" : {
      "id" : 214,
      "name" : "Spain"
    },
    "managers" : [ {
      "id" : 676,
      "name" : "Ronald Koeman",
      "nickname" : null,
```

```
      "dob" : "1963-03-21",
      "country" : {
        "id" : 160,
        "name" : "Netherlands"
      }
    } ]
  },
  "away_team" : {
    "away_team_id" : 902,
    "away_team_name" : "Huesca",
    "away_team_gender" : "male",
    "away_team_group" : null,
    "country" : {
      "id" : 214,
      "name" : "Spain"
    },
    "managers" : [ {
      "id" : 1605,
      "name" : "Juan José Rojo Martín",
      "nickname" : "Pacheta",
      "dob" : "1968-03-23",
      "country" : {
        "id" : 214,
        "name" : "Spain"
      }
    } ]
  },
  "home_score" : 4,
  "away_score" : 1,
  "match_status" : "available",
  "match_status_360" : "available",
  "last_updated" : "2021-03-16T12:45:17.775814",
  "last_updated_360" : "2021-07-27T01:28:10.084649",
  "metadata" : {
    "data_version" : "1.1.0",
    "shot_fidelity_version" : "2",
    "xy_fidelity_version" : "2"
  },
  "match_week" : 27,
  "competition_stage" : {
    "id" : 1,
    "name" : "Regular Season"
  },
  "stadium" : {
    "id" : 342,
    "name" : "Camp Nou",
```

```
    "country" : {
      "id" : 214,
      "name" : "Spain"
    }
  },
  "referee" : {
    "id" : 2559,
    "name" : "Adrián Cordero Vega",
    "country" : {
      "id" : 214,
      "name" : "Spain"
    }
  }
}
```


Listing 2 – Exemple des données issues de matches.Json depuis l'API StatsBomb pour le premier match de championnat entre le FC Barcelone et Huesca lors de la saison 2020/2021. Données récupérées avec la fonction sb.matches(match_id = 3773369).


```
{
  "team_id" : 217,
  "team_name" : "Barcelona",
  "lineup" : [ {
    "player_id" : 4447,
    "player_name" : "Martin Braithwaite Christensen",
    "player_nickname" : "Martin Braithwaite",
    "jersey_number" : 9,
    "country" : {
      "id" : 61,
      "name" : "Denmark"
    },
    "cards" : [ ],
    "positions" : [ {
      "position_id" : 23,
      "position" : "Center Forward",
      "from" : "84:21",
      "to" : null,
      "from_period" : 2,
      "to_period" : null,
      "start_reason" : "Substitution - On (Tactical)",
      "end_reason" : "Final Whistle"
    } ]
  }, {
    "player_id" : 5203,
    "player_name" : "Sergio Busquets i Burgos",
    "player_nickname" : "Sergio Busquets",
    "jersey_number" : 5,
```

```
    "country" : {
      "id" : 214,
      "name" : "Spain"
    },
    "cards" : [ ],
    "positions" : [ {
      "position_id" : 9,
      "position" : "Right Defensive Midfield",
      "from" : "00:00",
      "to" : null,
      "from_period" : 1,
      "to_period" : null,
      "start_reason" : "Starting XI",
      "end_reason" : "Final Whistle"
    } ]
  } ]
}
```


Listing 3 – Exemple des données de composition d'équipe du FC Barcelone pour les joueurs Martin Braithwaite et Sergio Busquets issues de lineups.Json depuis l'API StatsBomb pour le premier match de championnat entre le FC Barcelone et Huesca lors de la saison 2020/2021. Données récupérées avec la fonction sb.matches(match_id = 3773369)["Barcelona"].

	match_id	match_date	competition	season	home_team	away_team	home_score	away_score	referee	home_managers	away_managers
0	3773369	2021-03-15	Spain - La Liga	2020/2021	Barcelona	Huesca	4	1	Adrián Cordero Vega	Ronald Koeman	Juan José Rojo Martín
1	3773387	2021-05-11	Spain - La Liga	2020/2021	Levante	Barcelona	3	3	José Luis Munuera Montero	Francisco José López Fernández	Ronald Koeman
2	3773372	2021-05-08	Spain - La Liga	2020/2021	Barcelona	Atlético Madrid	0	0	Antonio Miguel Mateu Lahoz	Ronald Koeman	Diego Pablo Simeone
3	3773695	2021-05-02	Spain - La Liga	2020/2021	Valencia	Barcelona	2	3	José María Sánchez Martínez	Javier Gracia Carlos	Ronald Koeman
4	3773586	2021-04-29	Spain - La Liga	2020/2021	Barcelona	Granada	1	2	Pablo González Fuertes	Ronald Koeman	Diego Martínez Penas
5	3773457	2021-05-16	Spain - La Liga	2020/2021	Barcelona	Celta Vigo	1	2	NaN	Ronald Koeman	Eduardo Germán Coudet
6	3773377	2020-12-19	Spain - La Liga	2020/2021	Barcelona	Valencia	2	2	Alejandro José Hernández Hernández	Ronald Koeman	Javier Gracia Carlos
7	3773656	2020-11-21	Spain - La Liga	2020/2021	Atlético Madrid	Barcelona	1	0	NaN	Diego Pablo Simeone	Ronald Koeman
8	3773565	2021-01-09	Spain - La Liga	2020/2021	Granada	Barcelona	0	4	Ricardo De Burgos Bengoetxea	Diego Martínez Penas	Ronald Koeman
9	3773386	2020-10-31	Spain - La Liga	2020/2021	Deportivo Alavés	Barcelona	1	1	NaN	Pablo Javier Machín Díez	Ronald Koeman
10	3773587	2020-10-17	Spain - La Liga	2020/2021	Getafe	Barcelona	1	0	César Soto Grado	José Bordalás Jiménez	Ronald Koeman
11	3773672	2020-10-04	Spain - La Liga	2020/2021	Barcelona	Sevilla	1	1	Jesús Gil Manzano	Ronald Koeman	Julen Lopetegui Argote
12	3773585	2020-10-24	Spain - La Liga	2020/2021	Barcelona	Real Madrid	1	3	Juan Martínez Munuera	Ronald Koeman	Zinédine Zidane
13	3773466	2020-10-01	Spain - La Liga	2020/2021	Celta Vigo	Barcelona	0	3	Carlos del Cerro Grande	Óscar García Junyent	Ronald Koeman
14	3773552	2021-01-03	Spain - La Liga	2020/2021	Huesca	Barcelona	0	1	Guillermo Cuadra Fernández	Miguel Ángel Sánchez Muñoz	Ronald Koeman
15	3773593	2020-09-27	Spain - La Liga	2020/2021	Barcelona	Villarreal	4	0	Guillermo Cuadra Fernández	Ronald Koeman	Unai Emery Etxegoien
16	3773660	2020-12-13	Spain - La Liga	2020/2021	Barcelona	Levante	1	0	Ricardo De Burgos Bengoetxea	Ronald Koeman	Francisco José López Fernández
17	3773661	2021-04-22	Spain - La Liga	2020/2021	Barcelona	Getafe	5	2	Jorge Figueroa Vázquez	Ronald Koeman	José Bordalás Jiménez
18	3773597	2021-03-21	Spain - La Liga	2020/2021	Real Sociedad	Barcelona	1	6	José Luis Munuera Montero	Imanol Alguacil Barrenetxea	Ronald Koeman
19	3773523	2020-12-16	Spain - La Liga	2020/2021	Barcelona	Real Sociedad	2	1	José María Sánchez Martínez	Ronald Koeman	Imanol Alguacil Barrenetxea
20	3773571	2020-12-22	Spain - La Liga	2020/2021	Real Valladolid	Barcelona	0	3	Mario Melero López	Sergio González Soriano	Ronald Koeman
21	3773497	2021-04-10	Spain - La Liga	2020/2021	Real Madrid	Barcelona	2	1	Jesús Gil Manzano	Zinédine Zidane	Ronald Koeman
22	3773631	2021-02-07	Spain - La Liga	2020/2021	Real Betis	Barcelona	2	3	NaN	Manuel Luis Pellegrini Ripamonti	Ronald Koeman
23	3773665	2021-03-06	Spain - La Liga	2020/2021	Osasuna	Barcelona	0	2	Guillermo Cuadra Fernández	Jagoba Arrasate Elustondo	Ronald Koeman
24	3773428	2020-12-05	Spain - La Liga	2020/2021	Cádiz	Barcelona	2	1	César Soto Grado	Álvaro Cervera Díaz	Ronald Koeman
25	3764661	2021-01-06	Spain - La Liga	2020/2021	Athletic Club	Barcelona	2	3	Carlos del Cerro Grande	Marcelino García Toral	Ronald Koeman
26	3773526	2021-02-13	Spain - La Liga	2020/2021	Barcelona	Deportivo Alavés	5	1	Jorge Figueroa Vázquez	Ronald Koeman	Abelardo Fernández Antuña
27	3773474	2021-04-05	Spain - La Liga	2020/2021	Barcelona	Real Valladolid	1	0	Santiago Jaime Latre	Ronald Koeman	Sergio González Soriano
28	3773625	2021-02-27	Spain - La Liga	2020/2021	Sevilla	Barcelona	0	2	Alejandro José Hernández Hernández	Julen Lopetegui Argote	Ronald Koeman
29	3773403	2021-01-31	Spain - La Liga	2020/2021	Barcelona	Athletic Club	2	1	Antonio Miguel Mateu Lahoz	Ronald Koeman	Marcelino García Toral
30	3773547	2020-11-29	Spain - La Liga	2020/2021	Barcelona	Osasuna	4	0	Antonio Miguel Mateu Lahoz	Ronald Koeman	Jagoba Arrasate Elustondo
31	3773415	2021-02-21	Spain - La Liga	2020/2021	Barcelona	Cádiz	1	1	Juan Martínez Munuera	Ronald Koeman	Álvaro Cervera Díaz
32	3764440	2021-02-24	Spain - La Liga	2020/2021	Barcelona	Elche	3	0	Isidro Díaz de Mera Escuderos	Ronald Koeman	Francisco Escriba Segura
33	3773689	2021-04-25	Spain - La Liga	2020/2021	Villarreal	Barcelona	1	2	Carlos del Cerro Grande	Unai Emery Etxegoien	Ronald Koeman
34	3773477	2020-11-07	Spain - La Liga	2020/2021	Barcelona	Real Betis	5	2	NaN	Ronald Koeman	Manuel Luis Pellegrini Ripamonti

Table A.1 – Table regroupant les informations de tous les matchs de la saison 2020/2021 de La Liga via l'Open Data de StatsBomb.

	player_id	player_name	player_nickname	jersey_number	country	cards	positions
0	4447	Martin Braithwaite Christensen	Martin Braithwaite	9	Denmark	[]	[{'position_id' : 23, 'position' : 'Center Forwa...
1	5203	Sergio Busquets i Burgos	Sergio Busquets	5	Spain	[]	[{'position_id' : 9, 'position' : 'Right Defensi...
2	5211	Jordi Alba Ramos	Jordi Alba	18	Spain	[]	[{'position_id' : 8, 'position' : 'Left Wing Bac...
3	5477	Ousmane Dembélé	None	11	France	[]	[{'position_id' : 23, 'position' : 'Center Forwa...
4	5487	Antoine Griezmann	None	7	France	[]	[{'position_id' : 20, 'position' : 'Left Attacki...
5	5492	Samuel Yves Umtiti	Samuel Umtiti	23	France	[]	[]
6	5503	Lionel Andrés Messi Cuccittini	Lionel Messi	10	Argentina	[]	[{'position_id' : 18, 'position' : 'Right Attack...
7	6590	Norberto Murara Neto	Neto	13	Brazil	[]	[]
8	6826	Clément Lenglet	None	15	France	[]	[{'position_id' : 5, 'position' : 'Left Center B...
9	6947	Miralem Pjanić	None	8	Bosnia and Herzegovina	[]	[]
10	8118	Frenkie de Jong	None	21	Netherlands	[]	[{'position_id' : 4, 'position' : 'Center Back',...
11	17304	Héctor Junior Firpo Adames	Junior Firpo	24	Spain	[]	[]
12	20055	Marc-André ter Stegen	Marc-André ter Stegen	1	Germany	[]	[{'position_id' : 1, 'position' : 'Goalkeeper', ...
13	21881	Sergino Dest	Sergiño Dest	2	United States of America	[]	[{'position_id' : 7, 'position' : 'Right Wing Ba...
14	22390	Francisco António Machado Mota de Castro Trincão	Francisco Trincão	17	Portugal	[]	[{'position_id' : 7, 'position' : 'Right Wing Ba...
15	24841	Ricard Puig Martí	Riqui Puig	12	Spain	[]	[{'position_id' : 20, 'position' : 'Left Attacki...
16	29256	Matheus Fernandes Siqueira	Matheus Fernandes	19	Brazil	[]	[]
17	30486	Pedro González López	Pedri	16	Spain	[]	[{'position_id' : 11, 'position' : 'Left Defensi...
18	32480	Ronald Federico Araújo da Silva	Ronald Araújo	4	Uruguay	[]	[{'position_id' : 4, 'position' : 'Center Back',...
19	39073	Moriba Kourouma Kourouma	Ilaix Kourouma	27	Guinea	[]	[{'position_id' : 20, 'position' : 'Left Attacki...
20	39159	Arnau Tenas Ureña	Arnau Tenas	36	Spain	[]	[]
21	43728	Óscar Mingueza García	Óscar Mingueza	28	Spain	[]	[{'position_id' : 3, 'position' : 'Right Center ...

Table A.2 – Table regroupant les informations de la composition d'équipe du FC Barcelone pour le match du 15/03/2021 entre Barcelone et Huesca via l'Open Data de StatsBomb.

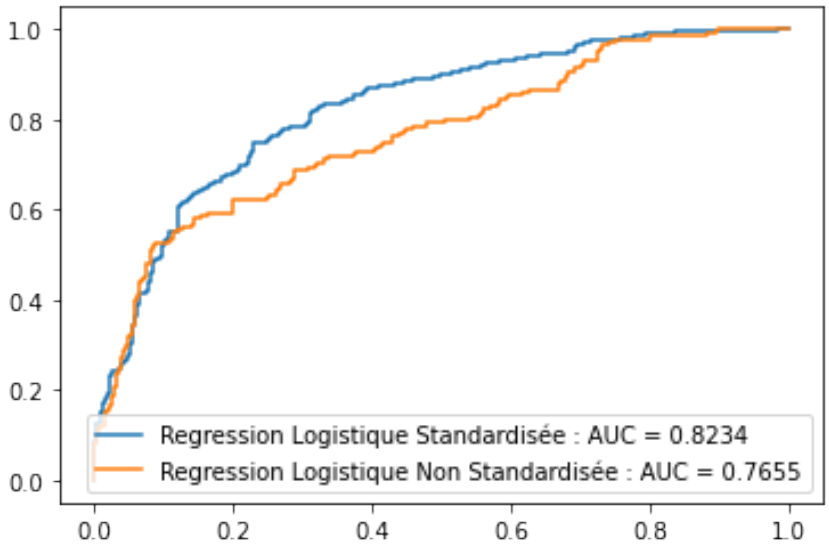


Figure A.1 – Comparaison des courbes ROC pour un modèle de régression logistique entre les données standardisées et non standardisées du dataset d'entraînement incluant les saisons 2017/2018, 2018/2019 et 2019/2020 avec une séparation aléatoire de 60% de données d'entrainement et 40% de données de test pour les deux modèles.

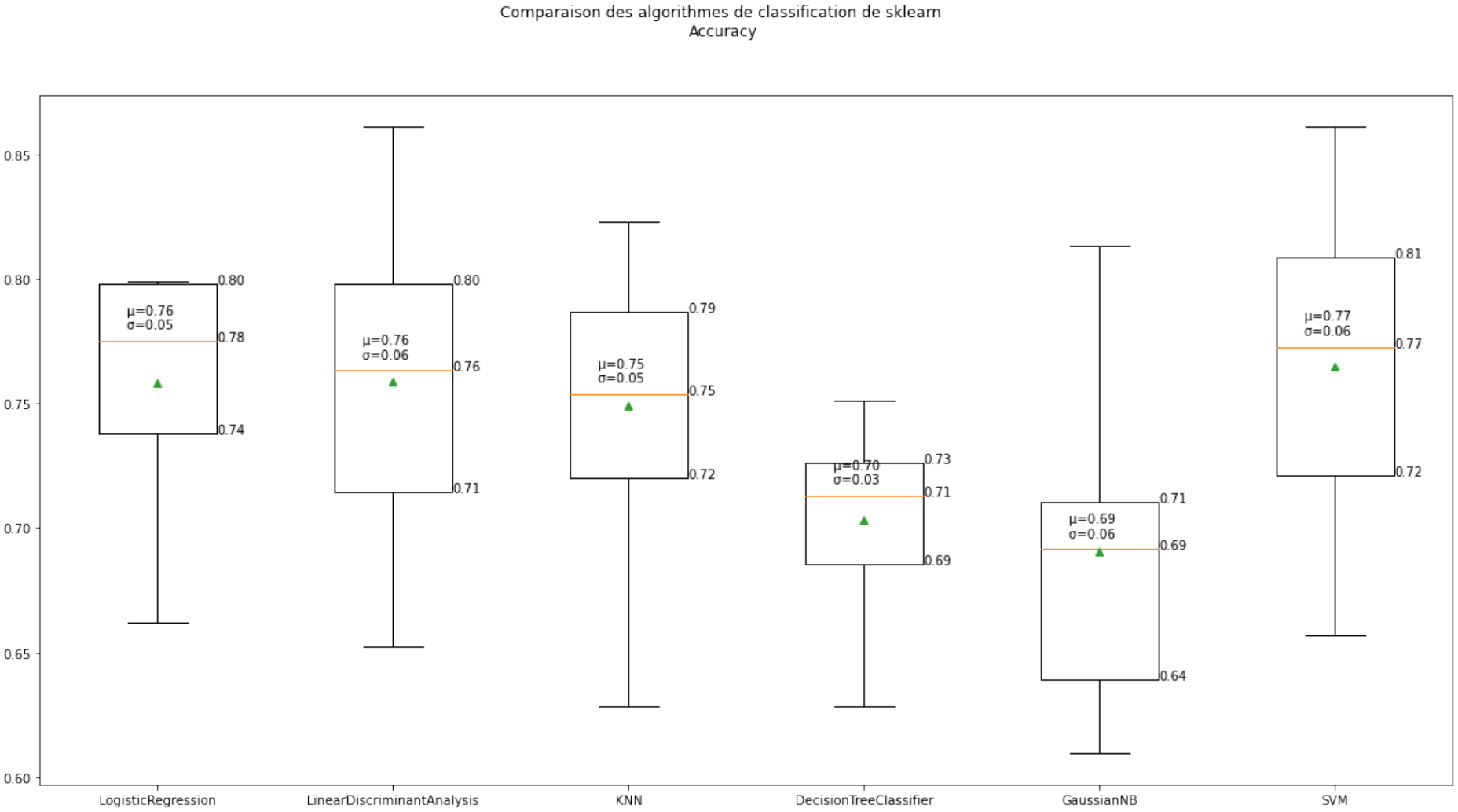


Figure A.2 – Boîte à moustache des performances des algorithmes de classification sur base de l'Accuracy.

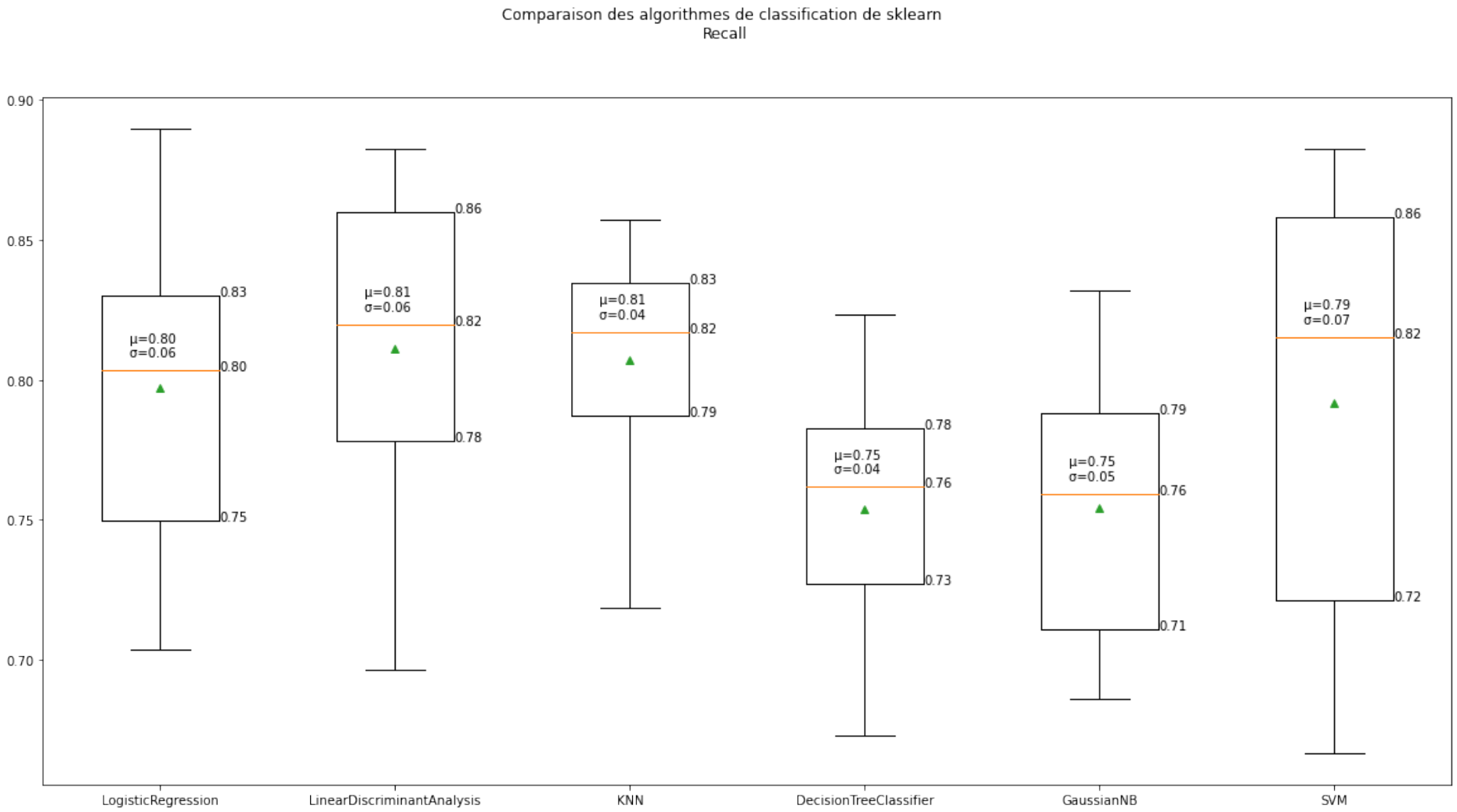


Figure A.3 – Boîte à moustache des performances des algorithmes de classification sur base du Recall.

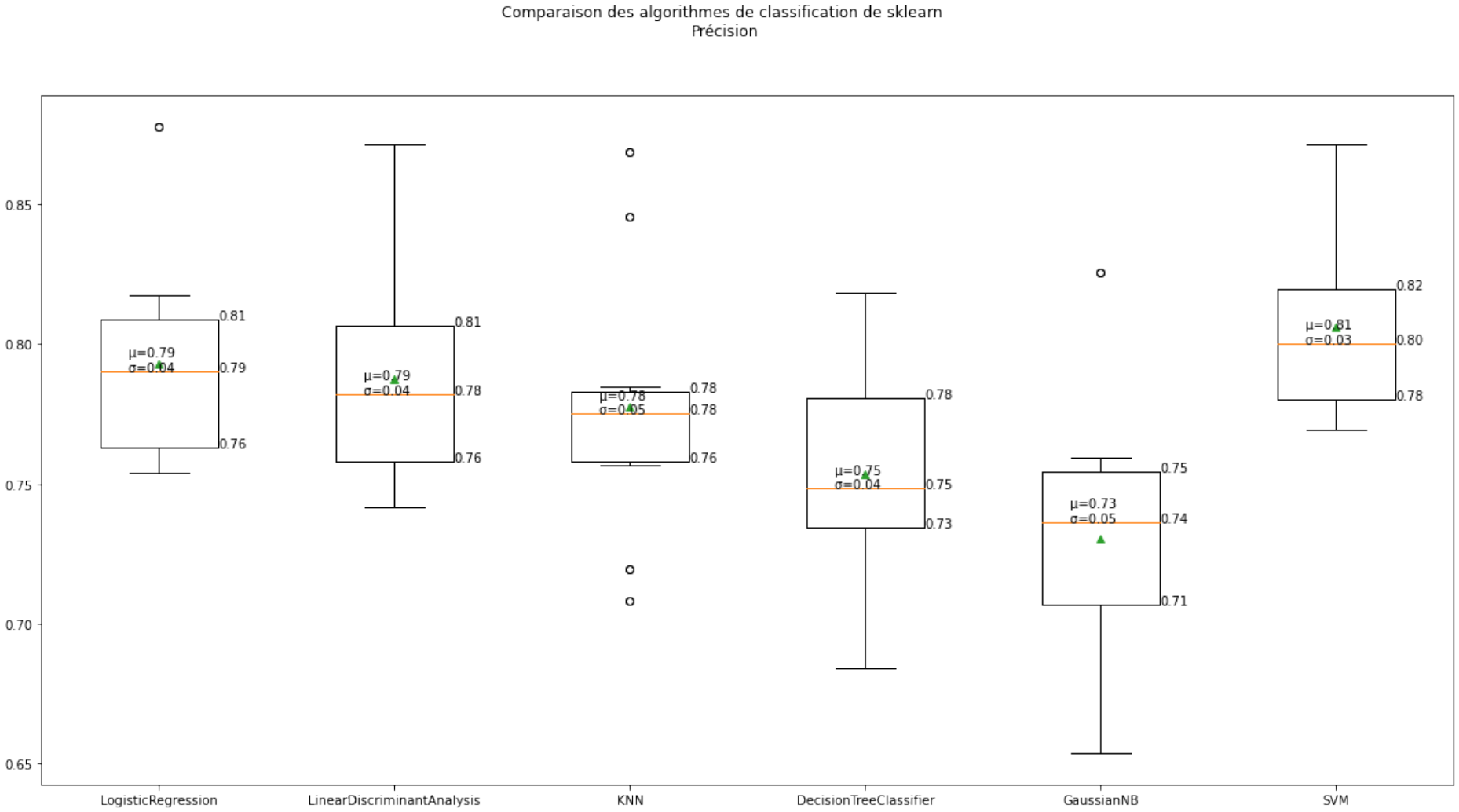


Figure A.4 – Boîte à moustache des performances des algorithmes de classification sur base de la Précision.

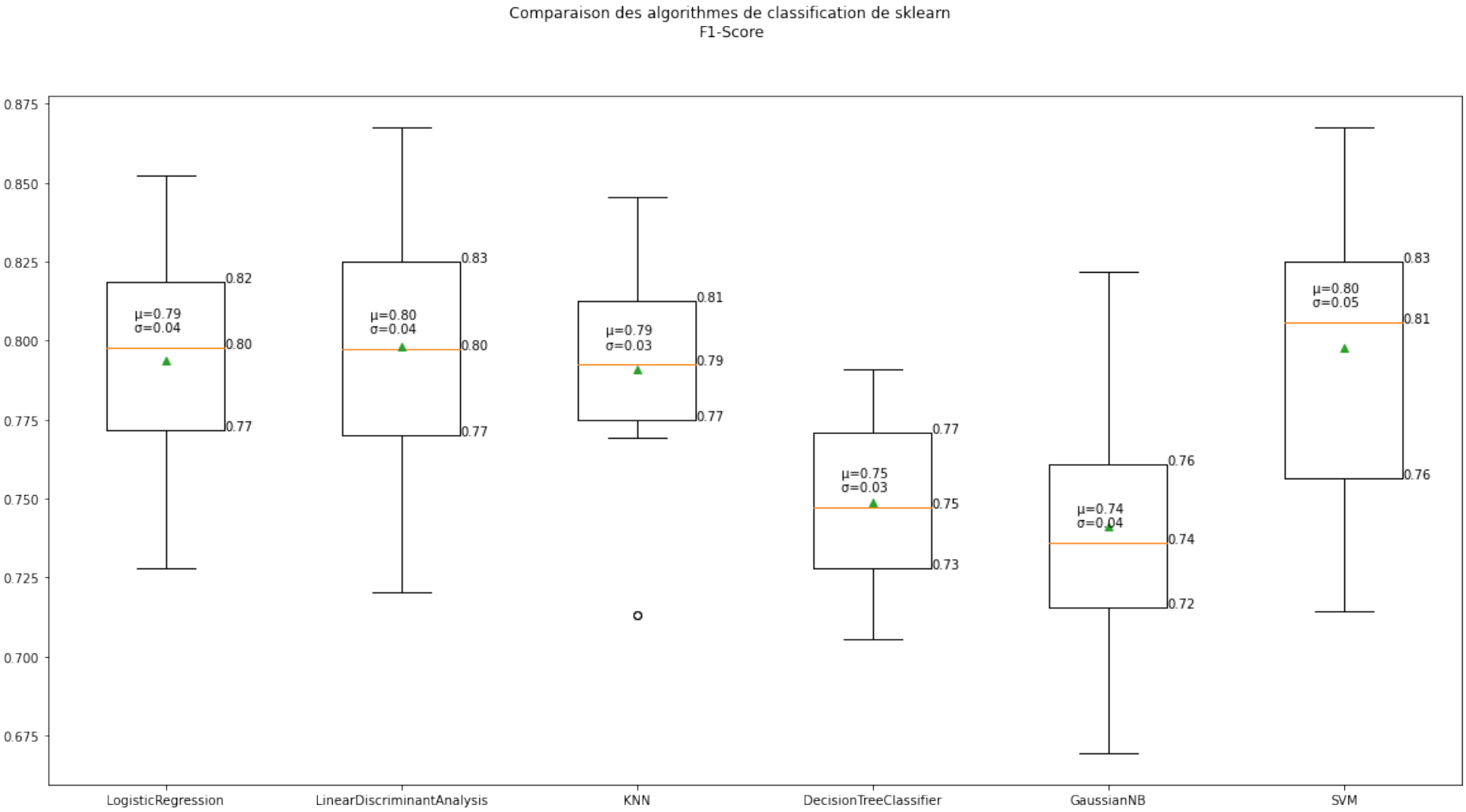


Figure A.5 – Boîte à moustache des performances des algorithmes de classification sur base du F1-Score.

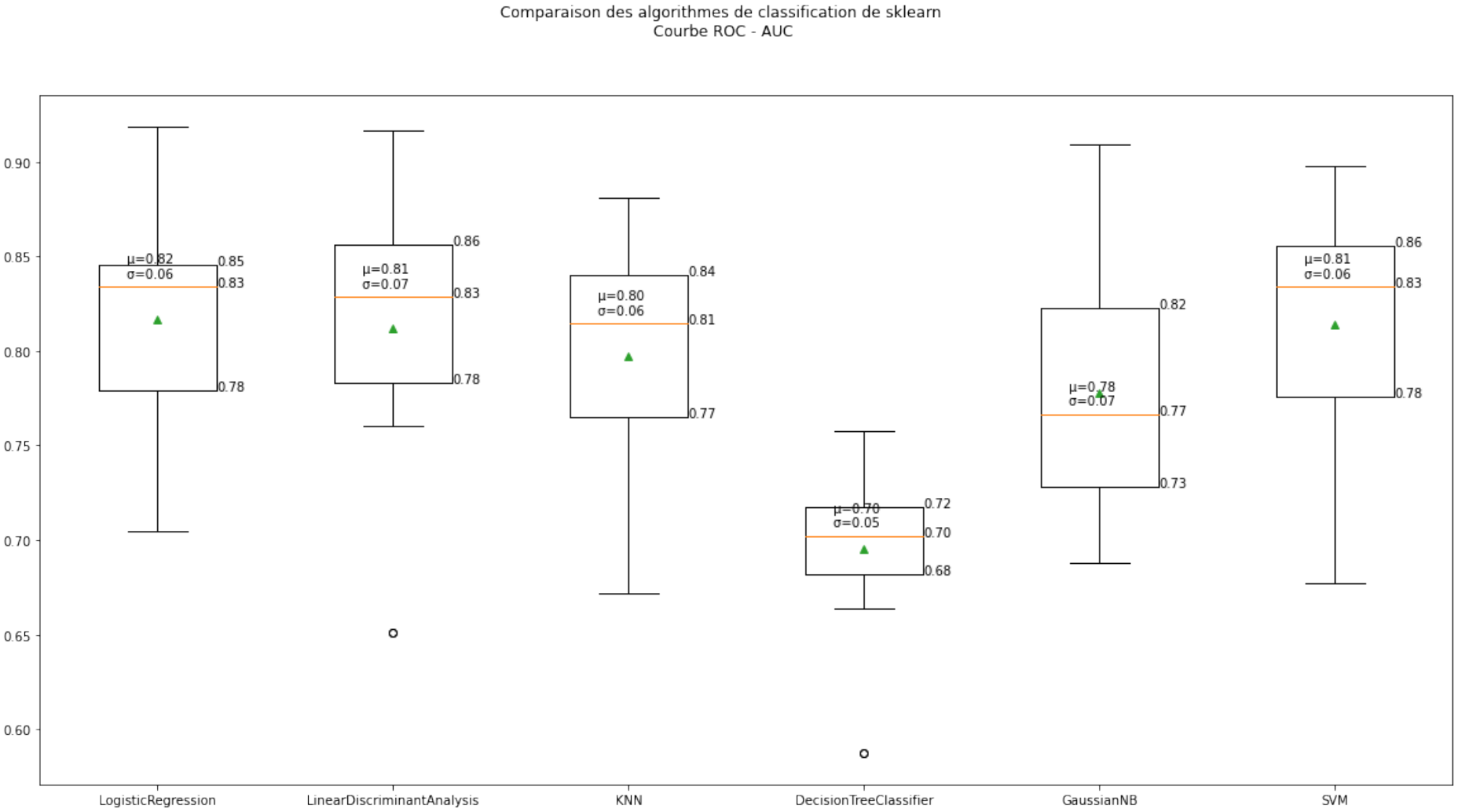


Figure A.6 – Boîte à moustache des performances des algorithmes de classification sur base de la ROC-Aera Under the Curve.

variable	Corrélation
overall	0.489696
international_reputation	0.450389
movement_reactions	0.446720
wage_eur	0.443352
mentality_composure	0.407435
attacking_short_passing	0.405265
age	0.404732
value_eur	0.398409
skill_ball_control	0.390870
release_clause_eur	0.389102
power_stamina	0.375779
passing	0.375719
skill_long_passing	0.364637
mentality_positioning	0.331357
mentality_vision	0.325518
sofifa_id	0.295387
player_id	0.292242
attacking_finishing	0.291147
club_jersey_number	0.285417
potential	0.280871
skill_curve	0.266837
shooting	0.263584
physic	0.263035
dribbling	0.257679
skill_fk_accuracy	0.254039
attacking_heading_accuracy	0.245516
attacking_crossing	0.233550
attacking_volleys	0.222749
selected_in_national_team	0.221787
mentality_penalties	0.219591
skill_dribbling	0.217326

variable	Corrélation
power_long_shots	0.201186
mentality_aggression	0.200746
power_strength	0.160793
weight_kg	0.148848
defending	0.123371
match_id	0.119734
mentality_interceptions	0.115804
weak_foot	0.115474
defending_sliding_tackle	0.114539
defending_standing_tackle	0.110908
defending_marking_awareness	0.095932
movement_agility	0.091650
power_shot_power	0.089550
movement_sprint_speed	0.061524
skill_moves	0.056226
height_cm	0.043020
pace	0.041591
club_contract_valid_until	0.031605
lost	0.022941
points	0.022689
movement_balance	0.019918
Win	0.019615
movement_acceleration	0.018092
away_score	0.015249
nationality_id	0.014336
home_team_id	0.010671
draw	0.007433
home_score	0.005519
away_team_id	0.004794
power_jumping	0.001210

Table A.3 – Liste des variables par ordre décroissant de leur corrélation avec la variable cible *played*.

Nom	Nombre de matchs prédits comme devant jouer	Nombre de matchs joués
L. Messi	35.0	35.0
Riqui Puig	35.0	11.0
A. Griezmann	35.0	33.0
Piqué	35.0	18.0
Jordi Alba	35.0	33.0
M. Pjanić	35.0	17.0
F. de Jong	35.0	34.0
Coutinho	35.0	11.0
Sergi Roberto	35.0	15.0
Sergio Busquets	35.0	33.0
Ansu Fati	35.0	7.0
Pedri	34.0	35.0
Matheus Fernandes	8.0	0.0
M. Braithwaite	8.0	26.0
O. Dembélé	2.0	27.0
Aleñá	0.0	2.0
S. Umtiti	0.0	11.0
Trincão	0.0	25.0
Junior Firpo	0.0	5.0
C. Lenglet	0.0	32.0
R. Araujo	0.0	21.0

Table A.4 – Tableau montrant le nombre de matchs pour lesquels le modèle classifie le joueur comme devant jouer comparé au nombre de matchs effectivement joué pour tous les joueurs du FC Barcelone lors de la saison 2020/2021.